
# UNIVERSITÀ DI PISA

**Dipartimento di informatica**
**Laurea in Informatica**

# Monitoraggio della guida di motocicli per mezzo di un sistema di visione multicamera con tecniche di fusione delle informazioni e loro rappresentazione virtuale per mezzo di un motore grafico 3D

**Relatore:**
**Prof. Davide Bacciu**
**Co-Relatori:**
**Marco Righi**
**Giuseppe Riccardo Leone**

**Presentata da:**
**Anthony Baiamonte**

**Sessione Autunnale**
**Anno Accademico 2022/2023**

# Indice

# Capitolo 1

# Introduzione

Il tirocinio si è svolto presso l'istituto di Scienza e Tecnologie dell'informazione del Consiglio Nazionale delle Ricerche (ISTI-CNR), da metà Marzo a fine Giugno 2023.
L'obiettivo del progetto è stato il monitoraggio della guida di motocicli per mezzo di un sistema di visione multicamera con tecniche di fusione delle informazioni e loro rappresentazione virtuale per mezzo di un motore grafico 3D.
Il tirocinio è stato realizzato nell'ambito di un progetto di ricerca del CNR dedicato alla progettazione e prototipazione di un sistema multicamera innovativo [1], basato su algoritmi di Computer Vision e Intelligenza Artificiale, in grado di riconoscere automaticamente, con grado di affidabilità variabile, le penalità previste durante la fase pratica degli esami per il conseguimento della patente di guida per motocicli. I dati raccolti dai singoli flussi video sono stati elaborati e fusi per ottenere un'informazione globale più completa ed affidabile. Particolare attenzione è stata dedicata al calcolo vettoriale delle accelerazioni del motoveicolo nei test in circuito chiuso. Inoltre, è stato realizzato un **Digital Twin** (gemello virtuale) del sistema per visualizzare in grafica 3D i dati elaborati e permettere l'analisi delle traiettorie effettuate dal motoveicolo in un contesto di prove ripetute.
Nel capitolo 2 si esporranno le basi di partenza del tirocinio, ovvero cosa era già presente prima di iniziare questo lavoro di tesi. Nel capitolo 3 si parlerà del background necessario a descrivere le metodologie e le tecnologie utilizzate in questa tesi. Nel capitolo 4 si elencheranno i software utilizzati per realizzare le varie funzionalità del sistema mettendo in evidenza le scelte effettuate, giustificando perché è stata scelta una particolare tecnologia piuttosto che un'altra. Nel capitolo 5 si descriverà il lavoro svolto durante il tirocinio mostrando tutti i test, gli esperimenti fatti e risultati ottenuti. Il capitolo 6 conclude la tesi riassumendo quanto si è realizzato e parlando dei possibili sviluppi futuri.

# Capitolo 2

# Basi di partenza del tirocinio

Lo stato dell'arte dei sensori dell'internet delle cose (IoT) relativo alla guida di ogni tipo di veicolo si sta significativamente evolvendo negli ultimi anni, trainato dall'ecosistema automobilistico e dalle applicazioni di guida autonome di nuova generazione, richiedendo un ammontare enorme di dati online per predire, calcolare e mantenere l'ottimale traiettoria e comportamento del motociclo/automobile. Tuttavia, nel contesto delle fasi di apprendimento umano e di allenamento alla guida di motocicli, questi strumenti non sono attualmente adeguati per fornire feedback istruttivi su manovre anomale, pericolose o inaccurate, visto che sono progettati per l'interazione con un sistema di calcolo piuttosto che con gli umani.
Dal 2006 (Direttiva 2006/126/EC del Parlamento Europeo sulle patenti di guida), sono state definite delle procedure di esame molto simili tra loro nelle nazioni dell'EU (ed anche nel mondo). Tuttavia, vi è una significativa mancanza di mezzi di verifica standard, misurabili e controllabili per gli esami di guida pratica dei motocicli.
Per esempio, nel 2021 nella provincia di Napoli (Italia), solo l'1.9% degli esami sono stati falliti, mentre più del 30% nella provincia da Cagliari (Italia). In Finlandia, sono necessari 3 anni per prendere la patente del motociclo, con un minimo di 37 ore di guida, mentre in altre nazione Europee non c'è nessun minimo ammontare definito. C'è bisogno di un sistema che fornisca mezzi affidabili e rigorosi di valutazioni, garantendo un trattamento giusto ed equo.

Il progetto di ricerca Artificial Intelligence - driven Riding Distributed Eye (AI-RIDE)[2], che è un caso d'uso del progetto europeo VEDLIot (H2020 ICT-56)[3], insegue questo obiettivo, presentando l'adozione di un sistema di AI accelerato, online ed incorporato nel contesto della formazione di motociclisti, mirando particolarmente a strumenti di verifica per le sessioni di Corsi di Guida Pratici (CGP) e Esame di Patente di Guida (EPG); di esso ne è stata fatta una dimostrazione pratica il 14 Giugno 2023 ([4]) a Pontedera (PI) presso i circuiti della autoscuola Gerardo in Via Tosco Romagnola 244, con la partecipazione di alcuni rappresentanti del Ministero delle Infrastrutture e dei Trasporti, della Motorizzazione Civile e delle autoscuole.
Tale sistema non ha intenzione di sostituirsi totalmente all'esaminatore, bensì vuole essere di supporto allo stesso nella valutazione del test di guida.
Il progetto si prefigge una innovazione dirompente nel contesto di apprendimento delle tecniche

di guida, andando significativamente oltre lo stato dell'arte degli strumenti attualmente utilizzati negli ecosistemi CPG ed EPG. In aggiunta, il progetto attiverà la definizione di un sistema di punteggio affidabile, oltrepassando il sistema corrente basato sul promosso/bocciato, fornendo allo stesso tempo indicazioni utili e misurabili di tipici errori d'esame e procedure di guida non sicure. Il risultato dell'esame di patente di guida dipende da diversi fattori:

- il tempo della performance
- la precisione della traiettoria
- la gestione della velocità,
- la postura del guidatore e la posizione del motociclo

Molti di questi fattori possono essere calcolati usando strumenti di analisi dei dati che si basano sui frame dei video catturati dalle telecamere installare lungo il circuito: grazie ad algoritmi di intelligenza artificiale si rileveranno i coni e il motociclo nell'immagine e grazie al sistema sviluppato in questa tesi se ne calcoleranno le posizioni dei coni, e la posizione, velocità e traiettoria della moto.
Per la legge italiana [5] l'esame della patente di guida consiste in 3 fasi: le prime due mirano a dimostrare le abilità di guida su un circuito sicuro chiuso al traffico, e solo dopo aver passato questa prova ne viene sostenuta una su strada pubblica. Questo lavoro di tesi si concentra sulle prime due fasi, con uno scenario già noto a priori dove è possibile posizionare delle telecamere e sensori adatti per la valutazione automatica delle abilità di guida del candidato. In particolare, questi test avverranno su dei percorsi predefiniti con misure standard delimitati da coni segnaletici (vedere figure 2.1, e 2.2):

1. il circuito Low Speed Balance (LSB) è più piccolo, ed è pensato per dimostrare le abilità in spazi stretti a basse velocità,
2. il circuito High Speed Agility (HSA), più grande del precedente, è pensato per dimostrare l'agilità del guidatore a velocità maggiori.

In entrambi i circuiti c'è uno slalom all'inizio, poi una curva ed infine un tratto rettilineo. Al termine del circuito HSA, il veicolo deve fermarsi in uno spazio specifico. I circuiti vanno percorsi rispettando i limiti di tempo imposti, e senza incorrere in penalità che altrimenti compromettono il successo dell'esame. Questi penalità sono:

- guida coordinata irregolarmente, dimostrando scarsa abilità;
- toccare uno o più coni;
- mettere un piede a terra;
- saltare un cono durante la fase dello slalom o uscire dal percorso;

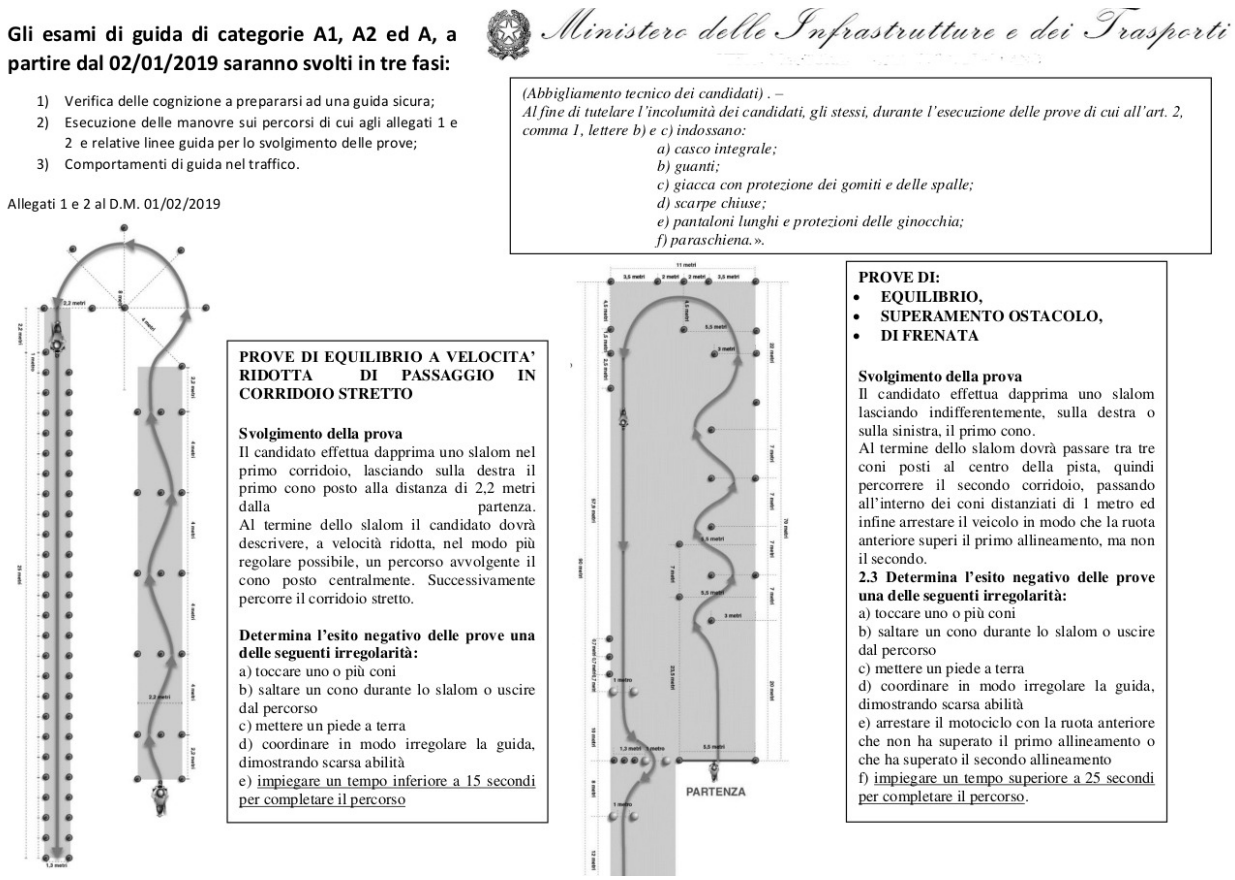



Figura 2.1: Dimensione dei due circuiti analizzati

- (Solo circuito HSA) fermare il motociclo con la ruota anteriore che non ha passato la prima linea oppure che ha passato la seconda;
- impiegare più di 25 secondi per completare il circuito HSA;
- impiegare meno di 15 secondi per completare il circuito LSB;

Si precisa che la parte sperimentale del lavoro svolto riguarda esclusivamente il circuito LSB, per adeguare il carico di lavoro alla durata del tirocinio. Lungo il perimetro della pista LSB sono piazzate due telecamere: una posizionata orizzontalmente alla pista, mentre l'altra la riprende verticalmente.
Elaborando i flussi video di queste due telecamere il compito del sistema sarà quello di essere di supporto all'esaminatore umano segnalando autonomamente una delle penalità elencate precedentemente, con un grado di affidabilità dettato dalla precisione dei dati e dalla difficoltà dell'evento da valutare.


Figura 2.2: I circuiti: a sinistra veduta aerea e frontale del circuito corto LSB (Low Speed Balance), nella parte destra veduta aerea di tutta la struttura con indicazione del circuito lungo HSA (High Speed Agility).

## 2.1 Architettura generale del sistema

Nelle seguente sezione si mostra l'architettura del sistema, progettata per riconoscere le penalità e determinare l'esito di un esame della patente di guida. Il team di lavoro è composto, oltre che dall'autore di questa relazione, dai due Co-Relatori Giuseppe Riccardo Leone e Marco Righi, e dal collega/tirocinante Davide Bulotta.
Ad oggi non si è a conoscenza di sistemi simili a quello su cui questa tesi basa il suo lavoro, ovvero nessuna soluzione completa basata sulla Computer Vision e sull'AI è disponibile sul mercato o è stata riportata in letteratura scientifica per automatizzare le procedure di allenamento e test per lezioni ed esami di guida di motocicli. L'unico annuncio riguarda Singapore [6] in cui si dichiara che alcune procedure generiche automatizzate basate sull'AI applicate a test di guida, dovrebbero essere disponibili nel 2024.
Durante la guida il motociclo è l'unico elemento in movimento di interesse nella scena. Tramite **Object Detector**, implementato con YOLOv5 (sezione 4.1, [7]), si riconosce l'elemento moto in ogni frame del video. Le informazioni ottenute grazie all'Object Detector vengono prima sottoposte ad una fase di fusione delle informazioni ed in seguito utilizzate come input per l'algoritmo di tracciamento composto da vari passaggi che in parte verranno analizzati nel dettaglio più avanti

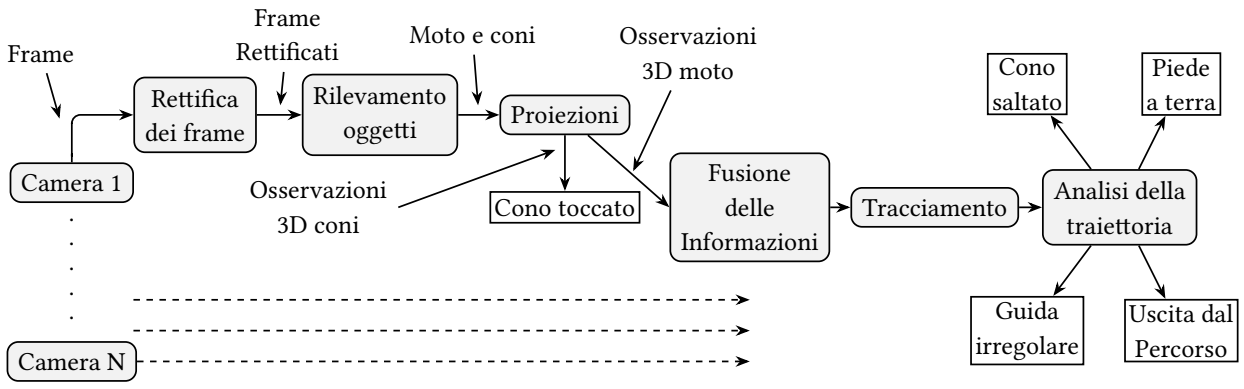



Figura 2.3: Architettura completa del sistema

in questa sezione, mentre i restanti verranno trattati nel capitolo 5.
L'algoritmo di tracciamento ha il compito di individuare il percorso che la moto sta facendo nella pista, memorizzando la sua posizione spaziale nel tempo. Infine, la traccia generata verrà poi sottoposta ad una fase di "analisi della traiettoria".
Tale flusso di elaborazione è mostrato in figura 2.3. Nelle capitolo 5 si illustreranno le idee di base e la logica dietro la segnalazione di ogni penalità connessa al test di guida. Per ognuna delle fasi presenti in figura 2.3 si avrà una sezione dedicata, nella quale si parlerà nello specifico del lavoro effettuato. Considerando la totalità delle fasi specificate in figura 2.3, in questo lavoro di tesi si descriveranno nel dettaglio quelle che vanno dalla fase "Proiezioni" fino alla fase finale di "Analisi della Traiettoria". Per correttezza di informazione si farà un accenno anche alla fase di "Rettifica dei frame" e di "Rilevamento oggetti" il quale non sono direttamente oggetto del tirocinio, per capire appieno il funzionamento del sistema.

## 2.2 Calibrazione della telecamera (Rettifica dei frame)

Alcune fotocamere stenopeiche introducono distorsioni significative nelle immagini. Si possono distinguere due tipi principali di distorsioni: la distorsione radiale e la distorsione tangenziale.
La **distorsione radiale** fa sì che le linee che dovrebbero essere dritte appaiano come curve. Tale fenomeno diventa sempre più intenso man mano che ci allontaniamo dal centro dell'immagine.

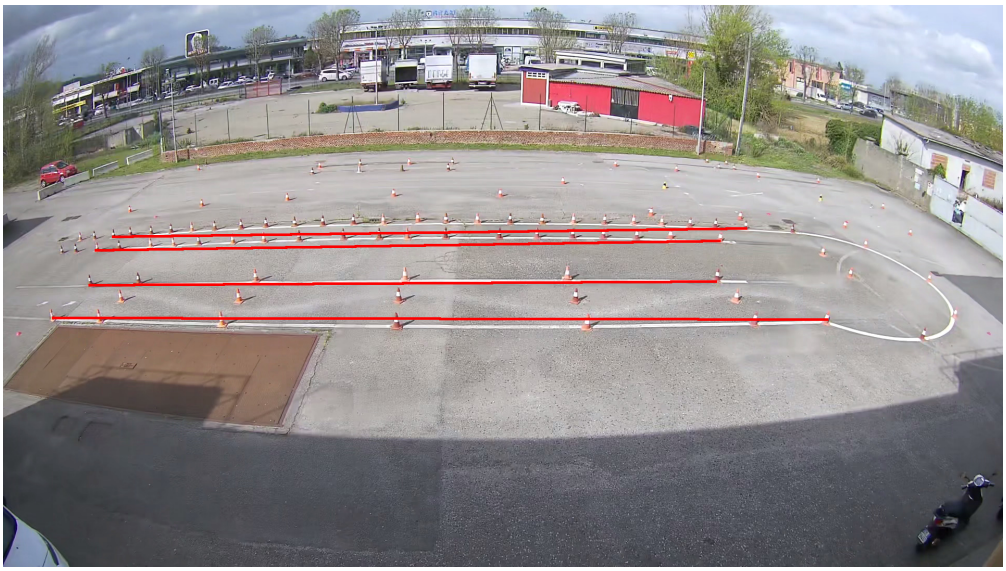


Figura 2.4: Frame della telecamera L2 (quella che riprende la pista verticalmente)

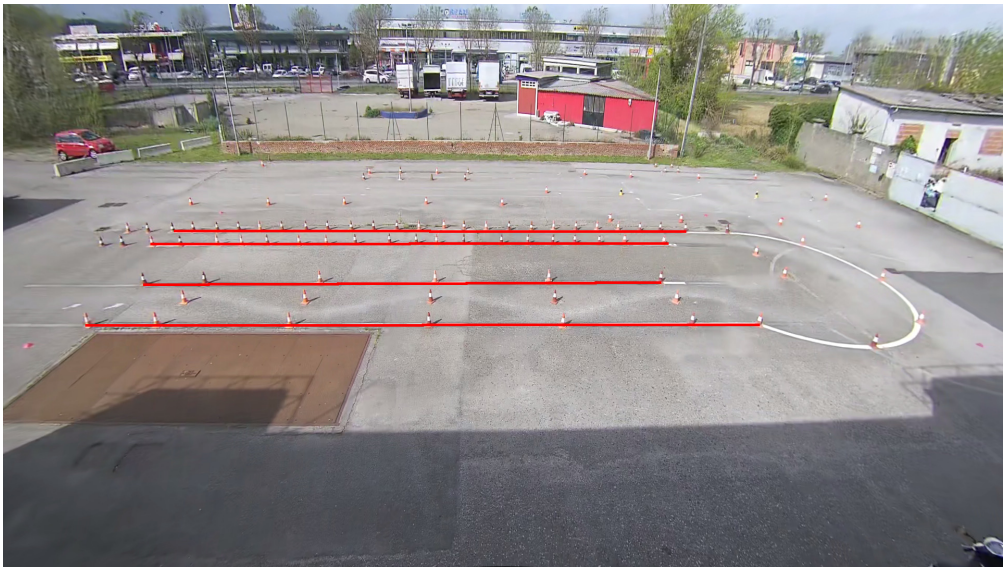


Figura 2.5: Frame della telecamera L2 (quella che riprende la pista verticalmente) dopo che è stata applicata la procedura di rettifica (undistortion)

Per esempio, nelle figure 2.4 e nella figura 2.5 si può vedere l'effetto di questa distorsione radiale, dove le linee che compongono la pista sono evidenziate in rosso. Si può vedere come nella figura 2.4 i bordi della pista non sono delle linee dritte e non combaciano con le linee rosse disegnate con una apposito editor di immagini. Questa distorsione radiale può essere rappresentata come descritto nelle equazioni 2.1 e in 2.2:

$$x_{distorted} = x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{2.1}$$

$$y_{distorted} = y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{2.2}$$

dove $(x, y)$ sono le coordinate di un punto nello spazio immagine rettificata, $r^2 = x^2 + y^2$ e $k_1, k_2, k_3$ sono tre **coefficienti detti di distorsione radiale**.
Allo stesso modo, la **distorsione tangenziale** avviene perché la lente per l'acquisizione di immagini non è allineata parallelamente al piano dell'immagine. Quindi alcune aree dell'immagine possono apparire più vicine di quanto lo siano realmente. La quantità di distorsione tangenziale può essere rappresenta come descritto nelle equazioni 2.3 e in 2.4.

$$x_{distorted} = x + [(2p_1 \cdot x \cdot y) + p_2 \cdot (r^2 + 2x^2)] \tag{2.3}$$

$$y_{distorted} = y + [p_1 \cdot (r^2 + 2y^2) + (2p_2 \cdot x \cdot y)] \tag{2.4}$$

dove $(x, y)$ sono sempre le coordinate di un punto nello spazio immagine rettificata, $r^2 = x^2+y^2$ e $p_1, p_2$ sono i **coefficienti di distorsione tangenziale**. Per rettificare le immagini bisogna trovare 5 parametri, conosciuti come **coefficienti di distorsione**:

$$\textit{Coefficienti di distorsione} = (k_1, \quad k_2, \quad p_1, \quad p_2, \quad k_3) \tag{2.5}$$

In aggiunta a questo, bisogna conoscere i parametri intrinsechi ed estrinsechi della telecamera. I **parametri intrinsechi** sono specifici della telecamera. Includono informazioni come la **distanza focale** $(f_x, f_y)$ e il **centro ottimale** $(c_x, c_y)$. Queste due informazioni vengono usate per creare la **matrice di camera** (2.6), il quale serve anch'essa per la rimozione della distorsione. La matrice di camera è unica per una specifica telecamera, cioè una volta calcolata, si può utilizzata su altre immagini ottenute dalla stessa telecamera. Essa è espressa come una matrice 3x3:

$$\textit{Matrice di camera} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

I **parametri estrinsechi** corrispondo ad una matrice di Rotazione **R**, e ad un vettore di traslazione **t** i quali proiettano le coordinate di un punto 3D in un altro sistema di coordinate.
Per trovare questi parametri, è necessario fornire alcune immagini campione di un pattern noto a priori (come ad esempio una scacchiera). Trovando alcuni punti specifici di cui conosciamo le posizioni relative (ad esempio le intersezioni tra i quadrati neri della scacchiera). Per avere dei buoni risultati è necessario avere almeno 10 immagini di test con la scacchiera in posizioni diverse.
Si considerino le immagini 2.6 e 2.7, i dati di input necessari per il processo di calibrazione della telecamera ([8], [9]) sono:

- l'insieme dei punti 3D nel mondo reale,
- le corrispondenti coordinate 2D di questi punti nell'immagine.

I punti immagine 2D si ricavano dall'immagine stessa (sono le coordinate del punto di contatto tra due quadrati neri della scacchiera).

Per definire invece quali siano i punti 3D nel mondo reale si fa la seguente supposizione. Sappiamo che queste immagini sono ottenute da una telecamera statica e le scacchiere sono piazzate a differenti posizioni e orientamenti, quindi abbiamo bisogno di conoscere le coordinate $(X, Y, Z)$. Però per semplicità, possiamo dire che la scacchiera è stata tenuta ferma su un piano XY, (quindi Z=0 sempre) e la telecamera era mossa di conseguenza. L'ipotesi che Z=0 sempre ci permette di concentrarci solamente sui valori di X e Y. Adesso per questi valori possiamo semplicemente passare i punti come (0,0), (1,0), (2,0), ... i quali denotano le posizioni dei degli incroci tra i quadrati assumendo che ognuno di essi abbiamo lato di dimensione 1. In questo caso, i risultati che otterremo saranno nella scala delle dimensioni del quadrato della scacchiera. Ma se conoscessimo la grandezza del quadrato, (ad esempio 30mm), potremmo passare i valori come (0,0), (30,0), (60,0), ... . In questo modo otterremo i risultati in mm. I punti 3D sono chiamati **Punti oggetto**, mentre i punti 2D nell'immagine sono chiamati **Punti immagine**.

Figura 2.6: Immagine risultato della procedura di undistortion, come si può notare mettendola a confronto con la figura accanto 2.7)

Figura 2.7: Immagine non ancora sottoposta ad undistorsion però con riconoscimento del pattern della scacchiera, passaggio necessario per calcolare i coefficiente di distorsione

Quando si va a cercare il pattern della scacchiera, è necessario indicare anche che tipo di pattern stiamo cercando: griglia 8x8, 5x5 ecc. Normalmente una scacchiera $N$x$N$ ha un numero di angoli interni pari a $(N-1)$x$(N-1)$.

## 2.3 Rilevamento Oggetti

Il nostro obiettivo è quello di ricostruire la traiettoria percorsa dalla moto a partire dai dati estratti da un video. Cioè vogliamo ottenere una sequenza di terne $(xR, yR, t)$ che rappresentano delle coordinate nel mondo reale ad intervalli di $1/50sec$ (nel paragrafo 5 si spiegherà il perché di questa frequenza).
I video registrati dalle telecamere L2 e L1 rappresentano la nostra principale fonte di informazioni da cui andremo ad estrarre i dati, e questo viene fatto applicando un **Object Detector** al video. Questo strumento, sviluppato a partire dal software YOLOv5 (vedere sezione 4.1), è in grado

di riconoscere nel video sia l'elemento moto in movimento, sia i vari coni sparsi per la pista. Infatti, ad ogni frame entrambi gli elementi prima citati sono circondati da un **Bounding Box** (un semplice rettangolo che racchiude la moto al suo interno) 4.1). L'applicazione dell'Object

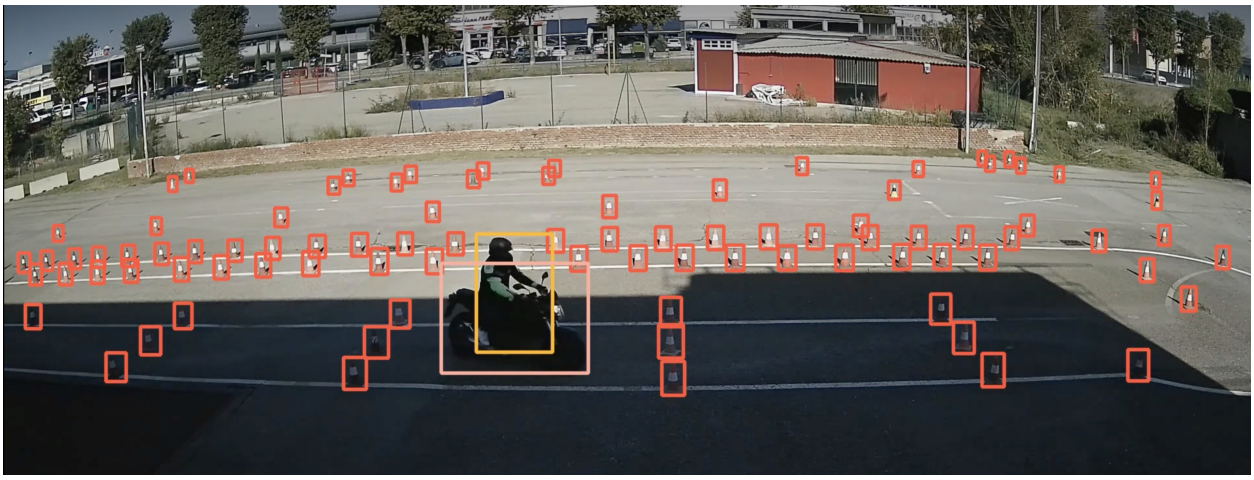


Figura 2.8: Esempio di un frame del video di partenza

Detector al video ci fornisce i seguenti output:

1. il video modificato in modo tale che sia la moto che i coni siano evidenziati da un Bounding box, come si vede in figura 2.8.

2. una serie di informazioni inserite in un file .txt, al cui interno viene specificata la posizione di ogni singolo bounding box che si trova nel frame.

In tabella 2.1 si osserva la conformazione predefinita di una parte dei file .txt generato dall'Object Detector, che è la seguente:

- in prima posizione troviamo sempre l'id della classe di riferimento del Bounding Box. Ogni oggetto identificato nel frame è associato ad una classe rappresentata da questo id. Nel nostro caso il numero 0 rappresenta un cono, mentre il numero 1 rappresenta la moto.

- dal secondo al quinto elemento vengono rappresentate due coppie (x,y), che sono le coordinate rispettivamente dello spigolo in alto a sinistra del Bounding Box e dello spigolo in basso a destra.

| id | x_bsx | y_bsx | x_tsd | y_tsd |
|---|---|---|---|---|
| 0 | 0.90651 | 0.594444 | 0.171875 | 0.444444 |
| 0 | 0.116927 | 0.559722 | 0.0161458 | 0.0416667 |
| 0 | 0.533333 | 0.518519 | 0.1666667 | 0.0416667 |
| 1 | 0.895052 | 0.514352 | 0.105729 | 0.137963 |
| 0 | 0.280469 | 0.606019 | 0.0182292 | 0.0472222 |

Tabella 2.1: Esempio della composizione di un file di output predefinito restituito da YOLOv5. I valori rappresentati sono in percentuale rispetto alla risoluzione di partenza del video analizzato, nel nostro caso 1920x1080.

Di base dovremmo avere un numero di file .txt pari al numero di frame che compongono il video, allora grazie alle coordinate espresse per ogni singolo bounding box, ad ogni frame sarà possibile recuperare le coordinare di ogni oggetto presente nello stesso (purché sia avvolto in un Bounding Box). Queste coordinate ottenute però sono relative allo spazio dell'immagine, e cioè sono delle coordinate in 2D (nell'immagine), che non possono ancora essere utilizzate così per come sono, perché necessitano ancora un po' di raffinamento che sarà effettuato nelle fasi successive alla corrente nel diagramma di flusso in figura 2.3. I file .txt che vengono restituiti da YOLOv5, si possono formattare in base alle esigenze, permettendo un'ampia possibilità di personalizzazione. I file che si utilizzano per rappresentare la traccia della moto e non sono quelli espressi in tabella 2.1, bensì saranno soggette ad una trasformazione ed in seguito ad una formattazione così da rendere semplice e veloce la loro interpretazione.
Per trasformazione e formattazione si intende che, a partire dai valori scritti nei file .txt (vedere tabella 2.1 per esempio) ci comporteremo in questa maniera:

- per convezione di sistema le righe che hanno come "id" hanno il valore 1 rappresentano le coordinate del Bounding Box della moto. Trasformazione e formattazione consistono nel prendere i valori dal secondo al quinto andare a calcolare il punto medio della base del rettangolo; questo avrà delle coordinate (x,y) nel frame in analisi. Ripetendo la procedura per ogni riga relativa alla classe moto tutte le coordinate (x,y) trovate verranno raggruppate in un unico file, come mostrato in tabella 2.2. In questo file si avrà un numero di righe che è pari al numero di volte che la moto viene rilevata all'interno del video.

- Le righe con valore 0 (sempre in tabella 2.1) rappresentano i coni. Per loro si farà qualcosa di simile a quanto fatto con la moto per quanto riguarda il calcolo delle coordinate nel frame in analisi. Anche per i coni manterremo un unico file con tutti i possibili cambiamenti dei coni. Un esempio di formattazione del file si può vedere in figura 2.3.

| n_frame | (x,y) |
|---|---|
| 200 | (451, 583) |
| 201 | (453, 584) |
| 202 | (456, 586) |
| 203 | (459, 587) |
| 207 | (469, 587) |
| 208 | (472, 588) |
| 209 | (476, 588) |

Tabella 2.2: Esempio di composizione del file che contiene le coordinate ottenute dal rilevamento della moto nel video

Analizzando la tabella 2.2, si nota che le coordinate del percorso della moto sono rappresentate da delle terne (n_framex,x,y). Assumiamo che la z sia sempre uguale a 0, perchè tutti gli eventi di interesse avvengono a livello del suolo. Per quanto riguarda la sua conformazione generale, possiamo osservare che una delle voci è **n-frame**: questa sta ad indicare il frame a cui si riferisce la posizione subito a seguire. Guardando l'esempio in tabella 2.2 notiamo che dal frame 203 si passa direttamente al frame 207, questo sta a significare che nei 4 frame intermedi la moto non è stata rilevata quindi sarebbe superfluo indicare delle posizioni quando in realtà non ci sono. Non possiamo far finta che la moto sia rimasta ferma durante quei 4 frame, perché essa starà continuando il suo percorso normalmente, per questo motivo in una delle fasi successive si applicherà un modello matematico (detto filtro di Kalman) che è in grado di predire le posizioni di un oggetto in movimento, e questo moto deve essere lineare o approssimabile come tale. Nella sezione successiva cercheremo di vedere se è legittimo fare questa assunzione oppure no.

| n_frame | id_cono | stato | (x,y) |
|---|---|---|---|
| 1 | 40 | 1 | (300,400) |
| 1 | 41 | 1 | (330,500) |
| 1 | 42 | 1 | (370.450) |
| 1 | 43 | 1 | (390,500) |
| 1 | 45 | 1 | (500,850) |
| 1 | 46 | 1 | (650,900) |
| 1 | 47 | 1 | (670,910) |

Tabella 2.3: Esempio di composizione del file che contiene le coordinate dei coni ottenute dal rilevamento nel video

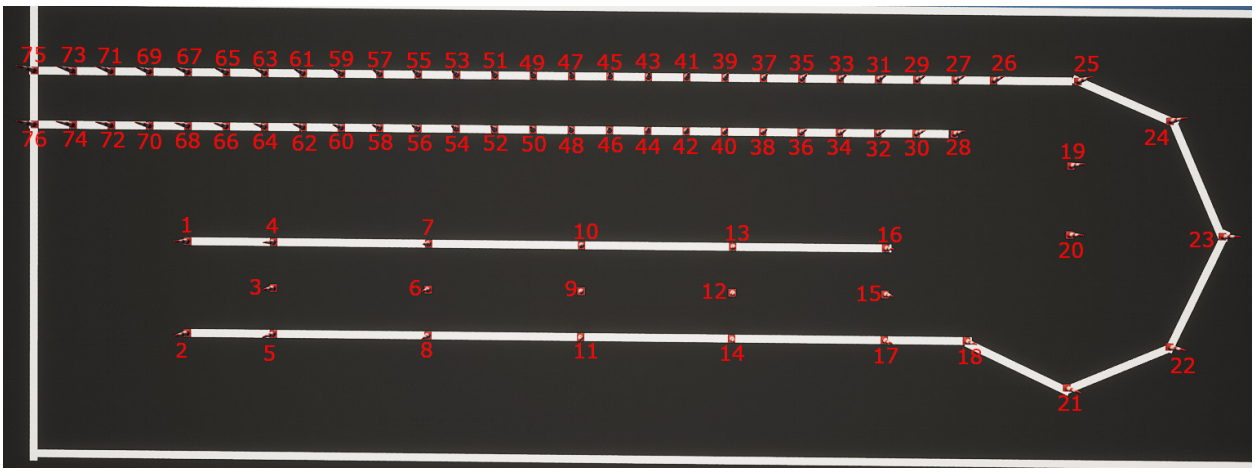



Figura 2.9: Mapping ID dei coni

Come possiamo vedere in tabella 2.3 notiamo che è sempre presente la voce **n-frame**, ma è anche presente una voce diversa rispetto al file che descriveva le coordinate della moto, l'id del cono (come si può vedere in figura 2.9), e lo stato. Il campo ID viene inserito perché la pista è composta da più coni, di conseguenza se più di uno di essi venisse toccato o cambiasse posizione per un motivo qualsiasi, dobbiamo essere in grado di capire quali di essi abbia subito uno spostamento. In questo modo, nel file in esame troveremo un riga relativa ad un cono e ad uno specifico frame se e solo se il cono indicato da quell'id ha subito uno spostamento in quel particolare frame altrimenti non sarà presente nessuna riga. La conformazione generale sarà la seguente:

- Una serie di righe con valore un determinato valore del frame e stato pari a 1. Queste righe definisco se un cono è presente nella pista oppure no. Se un cono con un determinato ID non dovesse avere una riga con stato pari a 1 allora vorrà dire che non è proprio presente nella pista.
- Una riga con un frame e ID qualsiasi ma con stato pari a 3 vorrà dire che nel frame di riferimento il cono non è stato avvistato.
- Quando un cono con un ID si trova nello stato 2 o 4 sta a significare che è stato visto rispettivamente mosso (posizione diversa da quella iniziale) oppure caduto (posizione diversa da quella iniziale ma anche capovolto) in un frame nello specifico.

# Capitolo 3

# Fondamenti metodologici

## 3.1 Statistica Descrittiva

Si parla di **statistica descrittiva** [10] quando i dati vengono analizzati senza fare assunzioni esterne all'insieme dei dati considerati. Lo scopo è quindi l'organizzazione dei dati in modo da evidenziarne la struttura e di rappresentarli in modo efficace.

### 3.1.1 Indici statistici

Supponiamo di avere un vettore $x = (x_1, ..., x_n) \in R^n$ di dati numerici. Gli indici statistici sono quantità numeriche che riassumono alcune proprietà significative della distribuzione di dati $(x_1, ..., x_n)$.
La media campionaria e la mediana campionaria descrivono dove si trova il "centro" della distribuzione dei dati:

**Definizione 3.1.1** (Media campionaria)**.** *La **media campionaria** è la media aritmetica dei dati,*

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{3.1}$$

**Definizione 3.1.2** (Mediana)**.** *La **mediana** è il dato $x_i$ tale che metà dei dati è minore o uguale a $x_i$ e l'altra metà maggiore o uguale (nel caso $n$ sia pari si può prendere la media aritmetica dei due valori centrali in questo senso).*

La media è solitamente preferita alla mediana. Tuttavia, la mediana è utile nel caso di dati molto asimmetrici e soprattutto è robusta rispetto alle code della distribuzione: in altre parole la media campionaria viene facilmente spostata da singoli dati molto piccoli o molto grandi, e questo non succede con la mediana.
La media campionaria può anche essere calcolata a partire dalle frequenze relative dei possibili

esiti: se questi ultimi sono $a_1, ..., a_M$ (ovvero ogni singolo dato $x_i$ è uguale a uno degli $a_j$).

$$\bar{x} = \sum_{j=1}^{M} a_j p(a_j, x), \quad p(a_j, x) = \frac{\#\{i : x_i = a_j\}}{n}. \tag{3.2}$$

dove appunto $p(a_j, x)$ è la frequenza relativa di $a_j$ nel campione $x$.
La varianza campionaria si usa per misurare la dispersione dei dati, cioè quanto i dati sono "sparsi" attorno alla media campionaria. Essa è infatti la media degli scarti quadratici dalla media campionaria $\bar{x}$.

**Definizione 3.1.3** (Varianza)**.** *La **varianza campionaria** oppure **varianza empirica**, rispettivamente*

$$var(x) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2, \quad var(x)_e = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2. \tag{3.3}$$

Come per la media campionaria, la varianza empirica (quindi anche quella campionaria) può essere espressa in termini delle frequenze relative: detti sempre $a_j,\ j = 1...M$, i possibili esiti e $p(a_j, x)$ la frequenza relativa di $a_j$ nel campione $x$, si ha

$$var(x)_e = \sum_{j=1}^{M} a_j^2\, p(a_j, x) - \bar{x}^2. \tag{3.4}$$

**Definizione 3.1.4** (Deviazione standard o scarto quadratico medio)**.** *La radice della varianza è chiamata scarto quadratico medio o deviazione standard (esiste sia empirica che campionaria): la indichiamo con $\sigma(x)$ (o $\sigma_e(x)$ quando è empirica) e si ha dunque $\sigma(x) = \sqrt{var(x)}$.*

Evidentemente diciamo che la varianza $(\sigma(x)^2 = 0) \iff (\forall i, j. x_i = x_j)$, cioè i dati sono tutti uguali.
Consideriamo adesso coppie di dati (o dati bivariati). I dati dell'indagine sono in questo caso un insieme di $n$ coppie di numeri $(x, y) = ((x_i, y_j), ..., (x_n, y_n)) \in R^{2xn}$. Ad esempio come nel nostro caso, $(x_i, y_j)$ può essere la posizione di un punto disegnato su una immagine 2D (che rappresenterà qualcosa di concreto). Ricordiamo che $\bar{x}$ e $\bar{y}$ sono le media campionarie di $x$ e $y$ e $\sigma(x)$ e $\sigma(y)$ sono le relative deviazioni standard campionarie.

**Definizione 3.1.5** (Covarianza)**.** *Si chiamano covarianza campionaria e covarianza empirica i numeri*

$$cov(x) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1}, \quad cov(x)_e = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{3.5}$$

**Definizione 3.1.6** (Coefficiente di correlazione)**.** *Supponiamo che $\sigma(x) \neq 0$ e $\sigma(y) \neq 0$: si chiama coefficiente di correlazione tra $x$ e $y$ il numero*

$$r(x, y) = \frac{cov(x, y)}{\sigma(x)\sigma(y)}. \tag{3.6}$$

E' possibile dimostrare come utilizzando nella formula 3.6 covarianza empirica e deviazione standard empirica la definizione non cambia. Una proprietà molto importate legata al coefficiente di correlazione è che $|r(x,y)| \leq 1$.

### 3.1.2 Retta di regressione

Intuitivamente, il coefficiente di correlazione misura il legame di natura lineare tra i dati $x$ e $y$: introducendo la *retta di regressione* stiamo quantificando questo concetto. L'idea è di approssimare i dati $y_i$ con una combinazione lineare affine $a + bx_i$, per farlo misuriamo la distanza dei dati da tale retta con i quadrati degli scarti: siamo dunque condotti a cercare i parametri $a,\ b$ calcolando

$$\min_{a,b\in R} \sum_{i=1}^{n} (y_i - a - bx_i)^2 \tag{3.7}$$

Questa quantità ha un unico minimo e vale il seguente risultato.

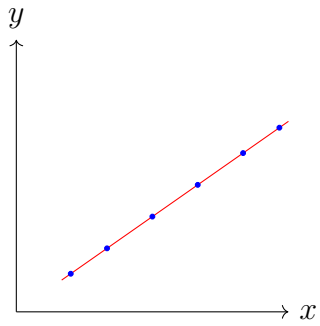
**Teorema 3.1.1.** *Supponiamo che $\sigma(x) \neq 0$ e $\sigma(y) \neq 0$. Il minimo al variare di $a,\ b \in R$ della quantità $\sum_{i=1}^{n}(y_i - a - bx_i)^2$ si ottiene scegliendo*

$$b^* = \frac{cov(x,y)}{var(x)}, \quad a^* = \bar{y} - b\bar{x} \tag{3.8}$$

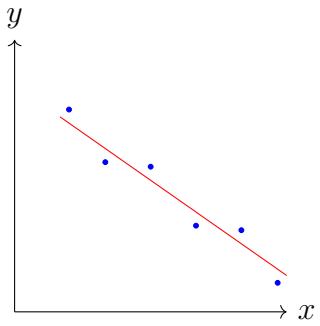
*e vale*

$$\min_{a,b\in R} \sum_{i=1}^{n} (y_i - a - bx_i)^2 = \sum_{i=1}^{n} (y_i - \bar{y})^2\ (1 - r(x,y)^2). \tag{3.9}$$

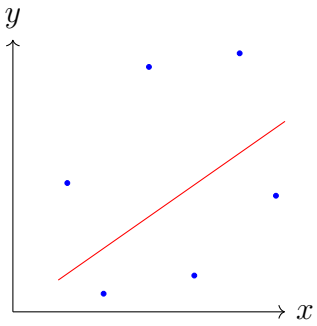
La retta $y = a^* + b^*x$ + chiamata retta di regressione: quanto più $r(x,y)$ è vicino a 1 in valore assoluto, tanto più i punti tendono ad essere allineati tale retta. In particolare vale che $|r(x,y)| = 1$ se e solo se tutti i punti si trovano su una stessa retta. Inoltre, $r(x,y)$ è positivo o negativo se il coefficiente angolare della retta è rispettivamente positivo o negativo. Se $r(x,y)$ è prossimo allo zero il teorema 3.1.1 mostra che non è possibile dare una buona approssimazione dei dati con una retta, nel senso che per ogni retta considerata esisteranno dei dati distanti da essa. Questo non esclude che possano esistere altre relazioni (approssimate) tra $x$ e $y$, ad esempio quadratiche o più in generale polinomiali.



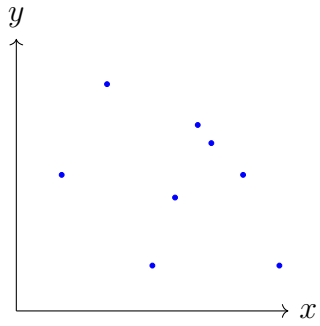

(a) r=1



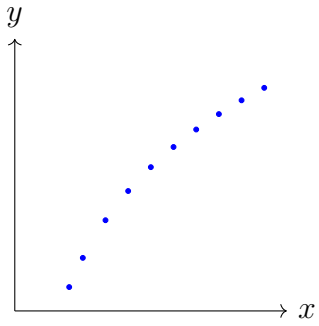

(b) r=0.9




(c) r=0.4




(d) r=0




(e) relazione non lineare

## 3.2 Fondamenti di Fisica

### 3.2.1 Velocità

La posizione di un oggetto ci dice dove esso si trova rispetto a un sistema di riferimento. La **velocità** ([11]) di un corpo ci dice con quale rapidità esso si muove e in quale direzione si dirige rispetto al sistema di riferimento, cioè è la rapidità con cui cambia la sua posizione, definita dal vettore posizione $\vec{\mathbf{r}}$.
Per definire formalmente la velocità partiamo dal concetto di **velocità media**. La velocità media $\langle\vec{\mathbf{v}}\rangle$ di un corpo in un intervallo di tempo compreso tra $t_i$ e $t_f$ è

$$\langle\vec{\mathbf{v}}\rangle = \frac{\vec{\mathbf{r}_f} - \vec{\mathbf{r}_i}}{t_f - t_i} = \frac{\Delta\vec{\mathbf{r}}}{\Delta t} \tag{3.10}$$

dove $\vec{\mathbf{r}_f}$ e $\vec{\mathbf{r}_i}$ sono i vettori posizione che individuano il corpo negli istanti $t_f$ e $t_i$ rispettivamente. Un simbolo tra parentesi acute (come il vettore velocità nell'equazione 3.10), rappresenta per convenienza il valore medio di una grandezza.
Abbiamo appena detto che la velocità media rappresenta la rapidità con cui un oggetto si muove e la direzione in cui si muove *durante un certo intervallo di tempo*. Di contro, la velocità istantanea, esprime la rapidità con cui un oggetto si muove e la sua direzione *in un certo istante di tempo*.

$$\vec{\mathbf{v}} = \lim_{\Delta t->0} \langle\vec{\mathbf{v}}\rangle = \lim_{\Delta t->0} \frac{\Delta\vec{\mathbf{r}}}{\Delta t} \tag{3.11}$$

La velocità è il limite a cui tende la velocità media quando l'intervallo di tempo tende a 0, e nella notazione del calcolo differenziale,

$$\vec{\mathbf{v}} = \frac{d\vec{\mathbf{r}}}{dt} \tag{3.12}$$

cioè la velocità di un punto materiale è la derivata prima rispetto al tempo del vettore posizione $\vec{\mathbf{r}}$ del punto stesso. Il modulo della velocità in coordinate cartesiane è così descritto:

$$|\vec{\mathbf{v}}| = v = \left|\frac{d\vec{\mathbf{r}}}{dt}\right| = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} \tag{3.13}$$

### 3.2.2 Accelerazione

L'**accelerazione** ([11]) di un corpo esprime la rapidità con la quale la sua velocità varia, sia in modulo che in direzione. Così come ci siamo serviti della velocità media per definire la velocità così ora utilizziamo l'accelerazione media per definire l'accelerazione.
L'accelerazione media $\langle\vec{\mathbf{a}}\rangle$ di un corpo nell'intervallo di tempo compreso tra $t_i$ e $t_f$ è data da

$$\langle\vec{\mathbf{a}}\rangle = \frac{\vec{\mathbf{v}}_f - \vec{\mathbf{v}}_i}{t_f - ti} = \frac{\Delta\vec{\mathbf{v}}}{\Delta t} \tag{3.14}$$

dove $\vec{\mathbf{v}}_f$ e $\vec{\mathbf{v}}_i$ sono le velocità negli istanti $t_f$ e $t_i$ rispettivamente.
Per determinare l'accelerazione, si deve trovare il limite cui tende l'accelerazione media quando l'intervallo di tempo tende a zero:

$$\vec{\mathbf{a}} = \lim_{\Delta t -> 0} \langle \vec{\mathbf{a}} \rangle = \lim_{\Delta t -> 0} \frac{\Delta \vec{\mathbf{v}}}{\Delta t} \tag{3.15}$$

Dal momento che

$$\lim_{\Delta t -> 0} \frac{\Delta \vec{\mathbf{v}}}{\Delta t} = \frac{d\vec{\mathbf{v}}}{dt} \tag{3.16}$$

definiamo l'accelerazione come

$$\vec{\mathbf{a}} = \frac{d\vec{\mathbf{v}}}{dt} \tag{3.17}$$

Poiché il vettore velocità è dato dalla derivata prima rispetto al tempo del vettore posizione (si veda L'Equazione 3.12), si può scrivere

$$\vec{\mathbf{a}} = \frac{d\vec{\mathbf{v}}}{dt} = \frac{d}{dt}\frac{d\vec{\mathbf{r}}}{dt} = \frac{d^2\vec{\mathbf{r}}}{dt^2}. \tag{3.18}$$

Tutte le formule descritte sia in questa sottosezione che nella precedente, sono solo delle formule generali, che possono essere sviluppate sia per i casi ad una, due o tre dimensioni. Lo stesso lavoro di sostituzione può essere fatto per i vettori velocità e accelerazione.

## 3.3 Campionamento di un segnale

**Segnale discreto [12].** Per **segnale discreto** si intende una successione di valori di una certa grandezza dati in corrispondenza di una serie di valori discreti nel tempo. In altri termini, è una funzione, o un segnale, con valori forniti in corrispondenza ad una serie di tempi scelti nel dominio dei numeri interi. Ciascun valore della successione è chiamato campione e si ottiene tramite l'operazione di **campionamento**.

**Campionamento [12].** Il **campionamento** è una tecnica che consiste nel convertire un segnale continuo nel tempo oppure nello spazio in un **segnale discreto**, valutandone l'ampiezza a intervalli temporali o spaziali regolari.

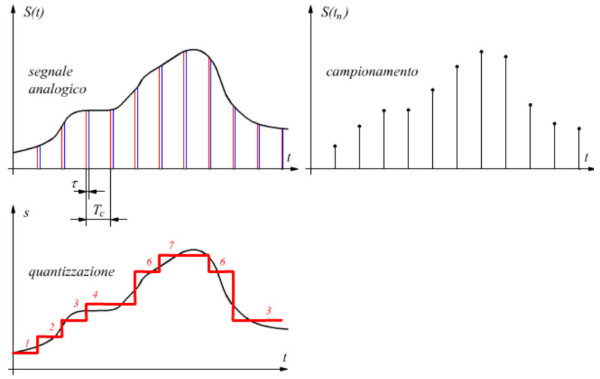



Figura 3.1: Rappresentazione del campionamento di un segnale. Il segnale continuo S(t) è rappresentato nel grafico in alto a sinistra, mentre i campioni discreti sono indicati del grafico in alto a destra.

In questo modo, a seguito di una successiva operazione di **quantizzazione** ([12]) e conversione, è possibile ottenere una stringa digitale che approssimi quella continua originaria.
Il campionamento consiste nel misurare e registrare il valore del segnale analogico in diversi istanti nel tempo o posizioni nello spazio. Il tempo T che intercorre tra una valutazione e l'altra si chiama $spaziale\ di\ campionamento$ o $intervallo\ temporale$. La $frequenza\ di\ campionamento$ $f_c = \frac{1}{T}$ è il reciproco dell'intervallo spaziale o temporale di campionamento, ed è anche il numero medio di campioni ottenuti al secondo.
Spesso si fa riferimento al campionamento su intervalli temporali, che è proprio quello che faremo noi.
Il teorema che stabilisce quale sia la frequenza minima di campionamento affinché il segnale analogico possa essere ricostruito a partire da quello discreto in input è il **teorema del campionamento** di Shannon:

**Teorema 3.3.1** (Teorema del campionamento di Shannon)**.** *Il **teorema del campionamento** di Shannon ([12]) afferma che la **frequenza di campionamento** $f_c$*

$$f_c = \frac{1}{T} \tag{3.19}$$

*deve essere almeno il doppio della frequenza massima $f_{max}$ con cui il segnale da campionare si presenta, cioè la frequenza più elevata fra le sue frequenza armoniche.*

$$f_c \geq 2 \cdot f_{max} \tag{3.20}$$

Se viene rispettata questa condizione è allora possibile ricostruire, con l'utilizzo di apposite funzioni interpolatrici, il segnale analogico senza perderne alcuna informazione. Qualora invece non venga rispettata tale condizione, si riscontra un effetto conosciuto come aliasing, che comporta una distorsione del segnale analogico ricostruito.
Generalmente per una buona e fedele ricostruzione del segnale analogico è richiesta una frequenza di campionamento che sia 5-10 volte maggiore alla frequenza massima contenuta nel segnale campionato, il motivo è che in quel caso si potrebbero utilizzare delle funzioni interpolatrici più semplici.

## 3.4 Omografia (Trasformazione Prospettica)

L'**Omografia** ([13]) anche nota come **trasformazione prospettica** o **collineazione**, è una qualsiasi mappatura $P^d \rightarrow P^d$ che è lineare nello spazio incorporato $R^{d+1}$. Cioè data una scala non nota, una omografia è scritta come

$$\mathbf{u}' \simeq \mathbf{Hu} \tag{3.21}$$

dove **H** è una matrice $(d+1)$x$(d+1)$.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.22}$$

La trasformazione mappa ogni tripletta di punti collineari in una tripletta di punti a loro volta collineari (da qui il suo nome di collineazione). Se $H$ è una matrice non-singolare (cioè $det(H) \neq 0$) allora punti distinti tra loro vengono mappati in punti distinti loro. Un esempio di immagine mappata con l'omografia 2D si può vedere in figura 3.4.

Figura 3.2: (a)

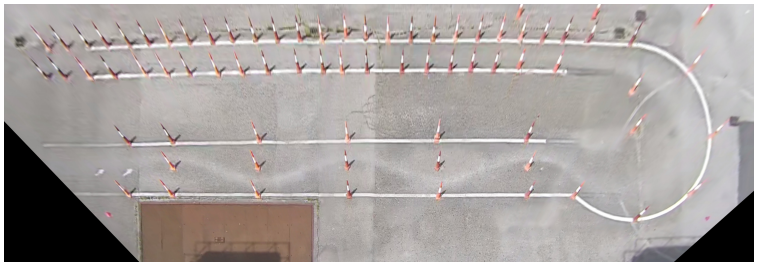


Figura 3.3: (b)

Figura 3.4: L'immagine 3.3 è il risultato della proiezione prospettica applicata all'immagine 3.2

Nella Computer Vision, ci sono tre semplici casi in cui si può applicare il concetto di omografia:

1. Il primo caso si ha quando vogliamo proiettare il piano di una immagine su un'altra superficie piana, come in figura 3.5.

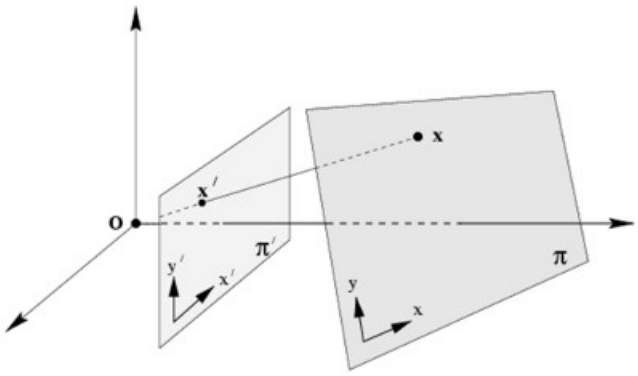



Figura 3.5

2. Il secondo caso si ha quando abbiamo una superficie piana vista da una camera e vogliamo applicare una proiezione in maniera tale che la superficie sia vista da un'altra camera in posizione diversa, come si può notare in figura 3.6. Metodo utilizzato per la rimozione o correzione della prospettiva.

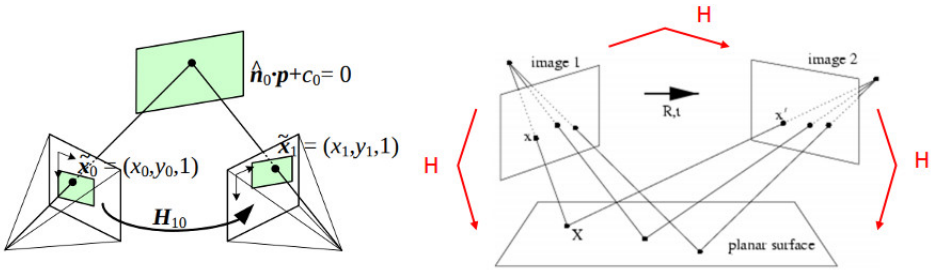



Figura 3.6

3. Terzo caso: una camera rotante attorno al proprio asse di proiezione, equivalente a considerare che i punti sono su un piano all'infinito (fig. 3.7). Un esempio di utilizzo potrebbe essere la cucitura di immagini panoramiche da diverse fotografie.

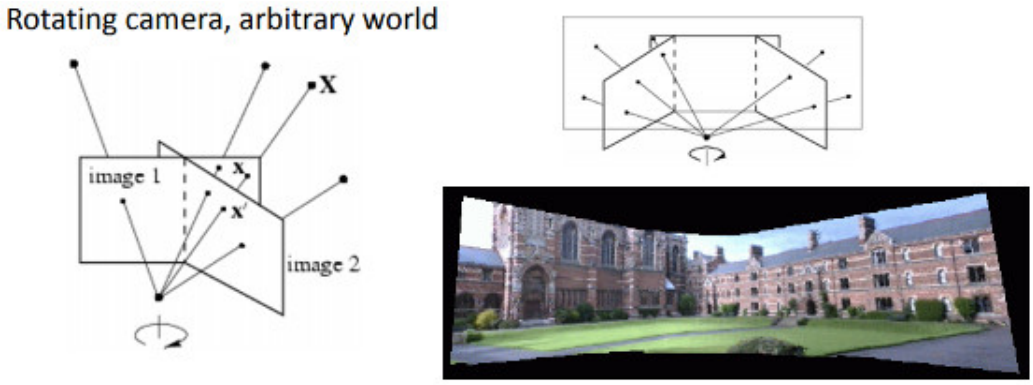



Figura 3.7

Per acquisire familiarità con le coordinate omogenee, è molto istruttivo mostrare in dettaglio come il punto 2D non omogeneo $[u, v]^T$ (ad esempio un punto in una immagine) è effettivamente mappato sul punto immagine non omogeneo $[u^{'}, v^{'}]^T$ usando la $H$ (3.22), e l'equazione (3.21). Con le componenti e la scala $\alpha$ scritte esplicitamente, l'equazione 3.21 diventa

$$\alpha \begin{bmatrix} u^{'} \\ v^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.23}$$

Scrivendo 1 nella terza coordinata del vettore $\mathbf{u}^{'}$, assumiamo che lo stesso non è un punto all'infinito, cioè si trova nella sua **forma canonica** e $\alpha \neq 0$, secondo quanto definito dalla teoria delle coordinate omogenee. Per calcolare il vettore $[u^{'}, v^{'}]^T$, è necessario eliminare la scala $\alpha$. Questo produce le seguenti espressioni

$$u^{'} = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + h_{33}} \tag{3.24}$$

$$v^{'} = \frac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{32}v + h_{33}} \tag{3.25}$$

familiari a coloro che non usano le coordinate omogenee per indicare vettori o punti nello spazio. Da notare che in relazione alle espressioni 3.24 e 3.25, l'espressione 3.21 è semplice, lineare, e può anche gestire il caso in cui $\mathbf{u}^{'}$ è un punto all'infinito. Questi sono i vantaggi pratici delle coordinate omogenee.

### 3.4.1 Stimare la matrice di Omografia dalle corrispondenze di punti

Un compito frequente nella Computer Vision 3D è quello di calcolare la matrice di omografia $H$ a partire dalle corrispondenze tra coppie di punti ([14]). Per **corrispondenze**, si intende un insieme $\{(u_i, u_i^{'})\}_{i=1}^m$ di coppie ordinate di punti così che ogni coppia corrisponda all'applicazione della trasformazione.
Per calcolare $H$, abbiamo bisogno di risolvere il seguente sistema omogeneo di equazioni lineari

$$\alpha_i \mathbf{u}_i^{'} = H\mathbf{u}_i, \quad i = 1, ..., m \tag{3.26}$$

per $H$ e la scala $\alpha_i$. Questo sistema ha $m(d+1)$ equazioni e $m + (d+1)^2 - 1$ incognite; ci sono $m$ valori di $\alpha_i$, e $(d+1)^2$ componenti di $H$. Così vediamo che $m = d + 2$ corrispondenze sono necessarie per determinare $H$ univocamente.
Alcune volte le corrispondenze formano una **configurazione degenere**, il quale significa che la matrice di trasformazione $H$ potrebbe non essere unica anche se $m \geq d + 2$.
Quando sono disponibili più di $d + 2$ corrispondenze, il sistema di equazioni 3.26 in generale non ha nessuna soluzione a causa del rumore ottenuto nel misurare le corrispondenze. Così, il semplice compito di risolvere un sistema di equazioni lineare diventa il problema più difficile di **stima ottimale** dei parametri di un modello parametrico. Cioè, non si risolve più l'equazione

3.26, ma minimizziamo un opportuno criterio, derivato da considerazioni statistiche. I metodi di stima che descriveremo brevemente adesso non si limitano unicamente all'omografia; sono dei metodi generici applicabili a diversi compiti nella Computer Vision 3D. Tra questi abbiamo: la **stima per massima verisimiglianza**, la **stima lineare** e la **stima robusta** (per maggiori dettagli si rimanda alla bibliografia [14]).

## 3.5 Filtro di Kalman

Per la statistica ed la teoria del controllo, il filtro di Kalman ([15], [16], [17]), anche noto come stimatore quadratico lineare (LQE), è un algoritmo che usa una serie di misure osservate nel tempo, includendo rumori statistica ed altre inaccuratezze, e produce stime di variabili non note che tendono ad essere più accurate di quelle basate solo su una singola misurazione, stimando una distribuzione di probabilità congiunta sulle variabili per ciascun periodo di tempo.
Il filtro di Kalman ha numerose applicazioni tecnologiche. Un'applicazione comune è quella per la guida, navigazione e controllo di veicoli, in particolare aerei, veicoli spaziali e navi in movimento. Inoltre, questo filtro è applicato molto nell'analisi delle serie temporali usate per argomenti come l'elaborazione dei segnali. Il filtro di Kalman è anche uno degli strumenti principali nella pianificazione e controllo del movimento robotico, e può anche essere utilizzato per l'ottimizzazione della traiettoria.
L'algoritmo funziona con un processo in due fasi. Nella fase di **predizione**, il filtro di Kalman produce stime delle variabili di stato correnti, insieme alla loro incertezza. Una volta osservato che il risultato della successiva misurazione (necessariamente corrotto da qualche errore, incluso del rumore casuale) è osservato, queste stime sono aggiornate usando una media pesata, dando maggior peso alle stime con maggiore certezza.
L'algoritmo è ricorsivo. Può operare in tempo reale, utilizzando solo le misure presenti in ingresso e lo stato precedentemente calcolato e la sua matrice di incertezza; non sono richieste altre informazioni relative al passato.
Il filtro di Kalman con le sue estensioni è stato fondamentale nell'implementazione dei sistemi di navigazione dei sottomarini con missili balistici nucleari della Marina degli Stati Uniti e nei sistemi di guida e navigazione dei missili da crociera. Le idee di Kalman sono state anche applicate al problema non lineare di stima della traiettoria per il programma Apollo con conseguente incorporazione nel computer di navigazione Apollo ([18]). Il filtro è anche utilizzato nei sistemi di guida e navigazione dei veicoli di lancio riutilizzabili e nei sistemi di controllo dell'assetto e di navigazione dei veicoli spaziali che attraccano alla Stazione Spaziale Internazionale ([19]).

### 3.5.1 Panoramica del calcolo

Il filtro di Kalman usa un **sistema dinamico**, dati i punti di controllo per quel sistema, e misurazioni sequenziali multiple (che derivano per esempio da un sensore) per dare una stima delle variabili del sistema (il suo **stato**) che è migliore della stima ottenuta utilizzando una sola misura. In quanto tale, è un comune algoritmo di fusione di sensori e fusione di dati.
Dati di sensori rumorosi, approssimazioni nelle equazioni che descrivono l'evoluzione del sistema e fattori esterni che non vengono presi in considerazione, limitano quanto bene sia possibile determinare lo stato del sistema. Il filtro di Kalman affronta in modo efficace l'incertezza dovuta ai dati di sensori rumorosi e, in una certa misura, a fattori esterni casuali. Questo filtro (fig. 3.8) produce una stima dello stato del sistema come la media degli stati del sistema predetti e le nuove misurazioni usando una media pesata. Lo scopo dei pesi è che valori con migliori (cioè più picco-

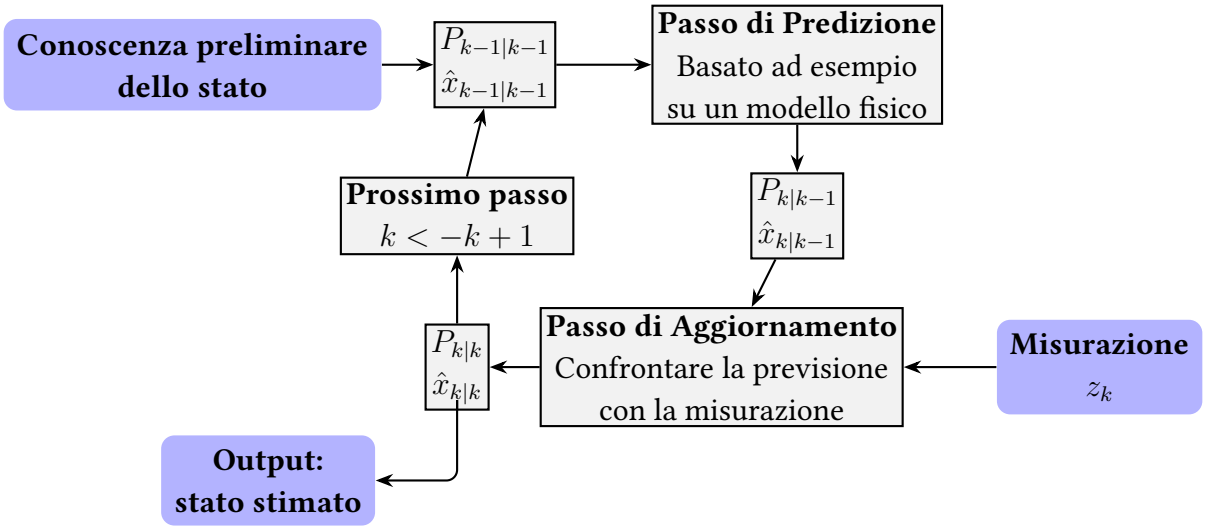



Figura 3.8: Il filtro di Kalman tiene traccia dello stato stimato del sistema e della varianza o incertezze della stima. La stima è aggiornata usando un modello di transizione dello stato e delle misurazioni. $x_{k|k-1}$ denota la stima dello stato del sistema al passo $k$ prima che la $k - esima$ misurazione $z_k$ sia presa in considerazione; $P_{k|k-1}$ è la corrispondete incertezza.

le) incertezze stimate sono più affidabili. I pesi sono calcolati dalla covarianza, che è una misura dell'incertezza stimata della predizione dello stato del sistema. Il risultato della nuova media pesata è una nuova stima dello stato che si trova tra lo stato predetto e quello misurato, e presenta un'incertezza stimata migliore rispetto a entrambi. Questo processo viene ripetuto ad ogni istante di tempo, con la nuova stima e la sua covarianza che informano la previsione utilizzata nella successiva iterazione. Ciò significa che il filtro funziona in modo ricorsivo e richiede solo l'ultima "migliore ipotesi", piuttosto che l'intera cronologia degli stati del sistema per calcolare il nuovo stato.
La valutazione della certezza delle misurazioni e la stima dello stato attuale sono degli elementi molto importanti del filtro. E' comune discutere la reattività del filtro in termini di **guadagno**. Il guadagno di Kalman è il peso dato alle misurazioni e alla stima dello stato corrente, e può essere "ottimizzato" in modo tale da ottenere una prestazione particolare. Con un guadagno alto, il filtro da più peso alla misurazioni più recenti, e così si conforma a loro in modo più reattivo. Con un guadagno basso, il filtro si conforma più strettamente alle previsioni del modello. Agli estremi, un guadagno elevato vicino a uno provocherà una traiettoria stimata più rumorosa, mentre un guadagno basso vicino a zero appianerà la presenza di rumore ma diminuirà la reattività del filtro. Quando si effettuano i calcoli effettivi per il filtro (come discusso sopra), lo stato stimato e la covarianza sono codificate in **matrici** a causa delle multiple dimensioni coinvolte in un singolo set di calcoli. Ciò consente la rappresentazione di relazioni lineari tra variabili di stato differenti (come la posizione, velocità e accelerazione) in uno qualsiasi dei modelli di transizione e covarianze.

### 3.5.2 Modello Formale

Il filtro di Kalman si basa su sistemi dinamici lineari discretizzati nel dominio del tempo. Lo stato del sistema target si riferisce alla configurazione del sistema di interesse, che è rappresentata come un vettore di numeri reali. Ad ogni incremento del tempo discreto, un operatore lineare è applicato allo stato per generare un nuovo stato, con un po' di rumore, e opzionalmente alcune informazioni dai controlli sul sistema se conosciuti. Poi, un altro operatore lineare misto ad altri rumori genera gli output misurabili (cioè le osservazioni) a partire dal vero stato.
Per far si che questo filtro possa essere utilizzato per stimare lo stato interno di un processo, avendo solamente una serie di osservazioni rumorose, è necessario astrarre dal processo impostando la seguente struttura, definita da una serie di matrici specificate per ogni passo temporale $k$:

- $\mathbf{F}_k$ la matrice di transizione dello stato. Supponendo che le seguenti equazioni descrivano il modello di movimento

$$\mathbf{p_k} = \mathbf{p_{k-1}} + \mathbf{v_{k-1}}\Delta t \tag{3.27}$$

$$\mathbf{v_k} = \mathbf{v_{k-1}} \tag{3.28}$$

  dove $\mathbf{p}$ è la posizione 3D dell'oggetto in movimento, $\mathbf{v}$ è il vettore velocità in 3D e $\Delta t$ è l'intervallo di tempo tra una misurazione e la successiva. La posizione $p_k$ e la velocità $v_k$ vengono chiamate **stato** del sistema. Da qui si ottiene **l'equazione di transizione dello stato**

$$\mathbf{x_k} = \begin{bmatrix} p_k \\ v_k \end{bmatrix} = F_k \begin{bmatrix} p_{k-1} \\ v_{k-1} \end{bmatrix} \tag{3.29}$$

  dove $\mathbf{F_k}$ ha la seguente forma.

$$\mathbf{F_k} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

- $\mathbf{H}_k$ la matrice di osservazione, ha il compito di mappare lo stato corrente sullo spazio di osservazione. La forma tipica di $\mathbf{H_k}$ è la seguente

$$\mathbf{H_k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.31}$$

  quando moltiplichiamo $\mathbf{H_k}$ per lo stato di stato $\mathbf{x_k} = \begin{bmatrix} p_k \\ v_k \end{bmatrix}$, stiamo semplicemente mantenendo i termini di posizione e scartando quelli di velocità.

- $\mathbf{Q}_k$ la matrice del rumore di processo. Ci possono essere dei fenomeni oltre il nostro controllo che affliggono lo stato dell'oggetto in movimento. Queste incertezze sono modellate dalla matrice $\mathbf{Q}_k$.

- $\mathbf{R}_k$ la matrice del rumore di osservazione;

- Lo stato dell'oggetto in movimento può anche essere influenzato da alcuni input di controllo, ad esempio l'acceleratore di una moto, oppure un sistema di guida autonomo di un drone per correggere la rotta. Questi controllo esterni sono modellati usando una matrice degli input di controllo $\mathbf{B}_k$ e $\mathbf{u}_k$ vettore di controllo, che rappresenta l'input nel modello di controllo. Ad esempio se $\mathbf{u_k}$ fosse il vettore di accelerazione 3D la forma della matrice $\mathbf{B_k}$ sarebbe la seguente

$$\mathbf{B_k} = \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & \frac{(\Delta t)^2}{2} & 0 \\ 0 & 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \tag{3.32}$$

Il modello del filtro di Kalman presuppone che il vero stato al passo $k$ sia ottenuto dallo stato al passo $k-1$ secondo la seguente formula:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{3.33}$$

dove:

- $\mathbf{F}_k$ è la matrice di transizione dello stato applicata al precedente stato $\mathbf{x}_{k-1}$;

- $\mathbf{B}_k$ è la matrice degli input di controllo applicata al vettore di controllo $\mathbf{u}_k$;

- $\mathbf{w}_k$ è un vettore che rappresenta il rumore di processo, che si presume derivi da una distribuzione multi-variata normale a media zero, $\mathcal{N}$, con covarianza, $\mathbf{Q}_k$ : $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$

Al passo $k$ una osservazione (o misurazione) $\mathbf{z}_k$ del vero stato $\mathbf{x}_k$ è ottenuta attraverso la seguente formula

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.34}$$

dove:

- $\mathbf{H}_k$ è la matrice di osservazione, che mappa lo spazio dei veri stati nello spazio degli stati osservati;

- $\mathbf{v}_k$ è il vettore che rappresenta il rumore di osservazione, che si presume sia rumore bianco Gaussiano a media zero con covarianza $\mathbf{R}_k$ : $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$

Lo stato iniziale, e i vettori del rumore $\{\mathbf{x}_0, \mathbf{W}_1, ..., \mathbf{W}_k, \mathbf{v}_1, ..., \mathbf{v}_k\}$ sono assunti come mutuamente indipendenti.
Come abbiamo già detto il filtro di Kalman è uno stimatore ricorsivo. Questo significa che solo lo stato stimato al precedente passo e la misura corrente sono necessarie per computare la stima dello stato corrente. Contrariamente alle tecniche di stima a gruppi, non è richiesta alcuna cronologia delle osservazioni e/o delle stime. In quanto segue, la notazione $\hat{\mathbf{x}}_{n|m}$ rappresenta la stima di $\mathbf{x}$ al passo $n$ date le osservazione fino al passo $m \leq n$.
Lo stato del filtro è rappresentato da due variabili:

- $\mathbf{x}_{k|k}$, la media stimata a posteriori dello stato al passo $k$, date le osservazioni fino al passo $k$ incluso;
- $\mathbf{P}_{k|k}$, la matrice di covarianza stimata a posteriori (una misura dell'accuratezza stimata della stima dello stato).

Il filtro di Kalman può anche essere espresso con una singola equazione; tuttavia, molto spesso è concettualizzato come due fasi distinte:

1. Fase di predizione
2. Fase di aggiornamento

La fase di predizione usa lo stato stimato al passo precedente per produrre una stima dello stato al passo corrente. Questa stima dello stato predetto è anche noto come stato stimato **a priori** perché, nonostante sia la stima dello stato al passo corrente, non include le informazioni sulle osservazioni del passo stesso.
Nella fase di Aggiornamento, l'**innovazione** (il residuo pre-fit), cioè la differenza tra l'attuale previsione a priori e le attuali informazioni di osservazione, viene moltiplicata per il guadagno ottimale di Kalman e combinato con la stima dello stato precedente per affinare la stima dello stato corrente. Questa stima migliorata basata sull'osservazione attuale è chiamata stima dello stato a *a posteriori*.
Tipicamente, le due fasi si alternano; la fase di predizione avanza lo stato fino alla successiva osservazione programmata, e la fase di aggiornamento che incorpora questa osservazione. Tuttavia, questo non è necessario, se un'osservazione non è disponibile per una qualunque ragione, la fase di aggiornamento può essere saltata e multiple fasi di predizione verranno eseguite. Allo stesso modo, se multiple osservazioni indipendenti sono disponibili allo stesso tempo, multiple fasi di aggiornamento possono essere eseguite (tipicamente con diverse matrici di osservazione $\mathbf{H}_k$).
Implementazioni pratiche del filtro di Kalman spesso sono molto difficili da fare a causa della difficoltà nell'ottenere delle buona stime delle matrici di covarianza del rumore $\mathbf{Q}_k$, e $\mathbf{R}_k$.

### 3.5.3 Fase di Predizione

- stima dello stato predetto (*a priori*):

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \mathbf{x}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{3.35}$$

dove $\mathbf{u_k}$ è il vettore di controllo, che rappresenta eventuali input di controllo nel moto dell'oggetto che stiamo tracciando. $\mathbf{B_k}$ è la matrice di controllo che è strettamente collegata al tipo di vettore $\mathbf{u_k}$ (per vederne un esempio si guardi la matrice 3.5.2). $\mathbf{x_{k-1|k-1}}$ rappresenta la conoscenza preliminare dello stato del sistema al passo k-1 considerando le misurazioni fino al passo k-1. $\mathbf{F_k}$ è la matrice di transizione di stato (di cui si può leggere un esempio in 3.30). Supponendo che le dimensioni di matrici e vettori siano corrette, le moltiplicazioni e la somma tra gli elementi appena descritti ci danno come risultato il vettore $\hat{\mathbf{x}}_{\mathbf{k|k-1}}$ che rappresenta la stima dello stato predetto al passo k, senza tenere conto della misurazione effettuata al passo k.

- stima della matrice di covarianza di stato predetta (*a priori*):

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathbf{T}} + \mathbf{Q}_k \tag{3.36}$$

dove $\mathbf{Q_k}$ è la matrice di covarianza del rumore di processo, il quale rappresenta le incertezze dovute a fenomeni che possono essere oltre il nostro controllo e che affliggono lo stato del sistema. $\mathbf{F_k}$ e $\mathbf{F}_k^T$ rappresenta rispettivamente la matrice di transizione di stato e la matrice di transizione di stato trasposta. $\mathbf{P}_{k-1|k-1}$ è la stima della matrice di covarianza di stato, che rappresenta l'accuratezza della stima dello stato, al passo k-1 tenendo conto di solo k-1 misurazioni. L'operazione ci da come risultato $\hat{\mathbf{P}}_{\mathbf{k|k-1}}$ che rappresenta la stima della matrice di covarianza di stato al passo k, senza tenere conto della misurazione effettuata al passo k.

### 3.5.4 Fase di Aggiornamento

- Misurazione (o innovazione) residua pre-aggiornamento:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \tag{3.37}$$

dove il vettore $\hat{\mathbf{x}}_{\mathbf{k|k-1}}$ rappresenta la stima dello stato predetto al passo k, senza tenere conto della misurazione effettuata al passo k. $\mathbf{H_k}$ è la matrice di osservazione, che mappa lo spazio dei veri stati nello spazio degli stati osservati. $\mathbf{z_k}$ è la misurazione (o osservazione) ottenuta al passo k, specificata dall'equazione 3.34. Il vettore $\tilde{\mathbf{y}}_\mathbf{k}$ rappresenta la misurazione residua prima dell'aggiornamento dello stato, cioè rappresenta la differenza tra la misurazione $\mathbf{z_k}$ e la moltiplicazione tra la matrice di osservazione e la stima dello stato predetto $\hat{\mathbf{x}}_{k|k-1}$.

- Stima della matrice di Covarianza della misurazione:

$$\mathbf{S}_k = \mathbf{H}_k \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^{\mathbf{T}} + \mathbf{R}_k \tag{3.38}$$

dove $\mathbf{R}_k$ è la matrice di covarianza del rumore di osservazione. $\mathbf{H_k}$ e $\mathbf{H_k^T}$ sono rispettivamente le matrice di osservazione dello stato e quella trasposta. $\hat{\mathbf{P}}_{\mathbf{k|k-1}}$ rappresenta la stima della matrice di covarianza di stato al passo k, senza tenere conto della misurazione effettuata al passo k (si faccia riferimento a 3.36). $\mathbf{S_k}$ rappresenta la stima della matrice di covarianza della misurazione, cioè ci dice di quanto variano le nostre misurazioni.

- guadagno di Kalman *ottimale*:

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{H}_k^{\mathbf{T}}\mathbf{S}_k^{-\mathbf{1}} \tag{3.39}$$

  Per $\hat{\mathbf{P}}_{\mathbf{k|k-1}}$ si faccia riferimento a 3.36. $\mathbf{H_k}$ rappresenta la matrice di osservazione dello stato. $\mathbf{S_k^{-1}}$ è l'inversa della stima della matrice di covarianza della misurazione $\mathbf{z_k}$. Più è grande $\mathbf{S_k}$ maggiore sarà la variabilità delle nostre misurazioni e minore sarà la nostra confidenza nelle stesse. Invece quando siamo confidenti nelle nostre misurazioni, cioè $\mathbf{S_k}$ è bassa, allora siamo sicuri che le informazioni ottenute siano abbastanza buone per aggiornare o modificare lo stato stimato del sistema, questo porta ad avere un **guadagno di Kalman alto**. Lo stesso si può dire di $\hat{\mathbf{P}}_{\mathbf{k|k-1}}$, questa rappresenta la variabilità dello stato stimato, se questa è grande, allora ci aspettiamo che lo stato cambia molto, quindi dobbiamo essere in grado di cambiare la nostra stima con nuove misurazioni, ottenendo così un alto guadagno. Al contrario, se questa variabilità è piccola, allora sappiamo che la stima dello stato non cambierà troppo, quindi non vogliamo alterarla troppo, cioè il guadagno è basso.

- stima stato del sistema aggiornata (*a posteriori*):

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{3.40}$$

  dove $\mathbf{y_k}$ è la misurazione residua pre aggiornamento, il quale viene moltiplicata per il guadagno ottimale di Kalman $\mathbf{K_k}$, il risultato poi viene sommato alla stima dello stato predetto $\hat{\mathbf{x}}_{\mathbf{k|k-1}}$. Queste operazioni danno come risultato lo **stima dello stato aggiornato** $\mathbf{x}_{k|k}$ al passo k tenendo conto anche della k-esima misurazione.

- stima della matrice di covarianza di stato aggiornata (*a posteriori*):

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\hat{\mathbf{P}}_{k|k-1} \tag{3.41}$$

  dove $\hat{\mathbf{P}}_{\mathbf{k|k-1}}$ è la stima della matrice di covarianza di stato predetta 3.36, $\mathbf{H_k}$ è la matrice di osservazione dello stato. $\mathbf{K_k}$ è il guadagno di Kalman e $\mathbf{I}$ è la matrice identità. $\mathbf{P_k}$ rappresenta la stima della matrice di covarianza di stato "aggiornata" e non più predetta perché teniamo conto della k-esima misurazione.

- misura residua post-aggiornamento:

$$\tilde{\mathbf{y}}_{k|k} = \mathbf{z}_k - \mathbf{H}_k\mathbf{x}_{k|k} \tag{3.42}$$

  dove $\mathbf{x}_{k|k}$ è la stima dello stato del sistema aggiornata al passo k tenendo conto anche della k-esima misurazione, $\mathbf{z_k}$ è la k-esima misurazione (osservazione). $\tilde{\mathbf{y}}_\mathbf{k}$ è la misurazione residua aggiornata facendo la differenza tra la k-esima misurazione $\mathbf{z_k}$ e la parte interessata della stima dello stato del sistema aggiornata $\mathbf{x}_{k|k}$.

## 3.6 Teoria delle Spline Monodimensionali

Nei sotto-campi informatici della progettazione assistita e della computer grafica, il termine spline ([20]) si riferisce più frequentemente ad una curva polinomiale (parametrica) a tratti ([21]). Le spline sono delle curve popolari in questi campi grazie alla semplicità della loro costruzione, dalla loro facilità e accuratezza di valutazione e della loro capacità di approssimare forme complesse attraverso il **curve fitting** (processo di costruzione di una curva a partire da una serie di punti) e la progettazione di curve interattive.

### 3.6.1 Uso delle spline

Il termine spline è usato per riferirsi ad una ampia classe di funzioni utilizzate in applicazioni che richiedono l'interpolazione e/o il livellamento dei dati. I dati possono essere unidimensionali o multidimensionali. Le *interpolation spline* sono normalmente definite come **minimizzatori** di opportune misure di rugosità dei dati, e sono soggette ai vincoli di interpolazione. Le *smoothing spline* possono essere viste come una generalizzazione delle *interpolation spline* dove le funzioni sono usate per minimizzare una combinazione pesata dell'errore di approssimazione al quadrato medio sui dati osservati e la misura di rugosità.
Come già accennato le spline monodimensionali sono utili in vari contesti grazie alle loro proprietà matematiche e alla capacità di approssimare curve o funzioni in modo flessibile. Ecco alcune ragioni per cui le spline monodimensionali sono ampiamente utilizzate:

- **Approssimazione dei dati**: consentono di approssimare insiemi di dati puntuali in modo accurato e regolare. Possono adattarsi alle fluttuazioni nei dati senza subire oscillazioni eccessive, garantendo una rappresentazione fluida e coerente.
- **Interpolazione**: possono essere utilizzate per interpolare un set di punti dati. Ciò significa che la curva spline passerà attraverso ogni punto di dati specificato, garantendo una rappresentazione precisa dei dati noti e consentendo di stimare valori tra i punti noti.
- **Grafica computazionale**: in questo campo le spline vengono spesso utilizzare per disegnare curve fluide e realistiche. Sono utili per rappresentare contorni di oggetti, tracciati di animazioni e altre rappresentazioni visive.
- **Design industriale e CAD (Computer-Aided Design)**: consentono di creare curve e superfici complesse che soddisfano requisiti specifici di forma e regolarità.
- **Modellazione matematica**: le spline possono essere utilizzate per approssimare funzioni complesse che descrivono dati sperimentali o fenomeni naturali.
- **Analisi dei dati**: utili per l'analisi e la rappresentazione dei dati. Utilizzate per creare curve di regressione flessibili che si adattano ai dati e consentono di identificare tendenze nascoste.

- **Controllo di movimento**: utilizzate spesso nei sistemi di controllo di movimento, come nel controllo di robot e macchine CNC. Consentono di generare traiettorie fluide e precise per il movimento di dispositivi meccanici.

In generale, le spline monodimensionali offrono un equilibrio tra flessibilità e regolarità, rendendole uno strumento prezioso in numerosi campi in cui è richiesta una rappresentazione accurata e fluida di curve e funzioni.

### 3.6.2 Definizione spline Monodimensionali

Grazie alla loro struttura semplice e alle proprietà di buona approssimazione, le spline sono ampiamente usate nella pratica per l'approssimazione di funzioni. Il grado di questa approssimazione dipende essenzialmente dal grado dei polinomi e dalla lunghezza dell'intervallo $[a, b]$ considerato. Poiché le operazioni di calcolo sui polinomi di alto grado comportano alcuni problemi solitamente si usano polinomi di basso grado. In questi casi, in modo tale da ottenere la precisione desiderata, è necessario limitarci ad operare su piccoli intervalli. A tal fine, l'intervallo originale $[a, b]$ viene diviso in sotto intervalli sufficientemente piccoli $\{[x_k, x_{k+1}]\}_{k=0}^{n}$ e poi si usano polinomi $p_k$ di bassi grado per le approssimazioni sui sotto intervalli $[x_k, x_{k+1}],\ k = 0, ...n$. Questa procedura produce una *funzione di approssimazione polinomiale definita a tratti* $\mathbf{s}(\mathbf{x})$,

$$s(x) \equiv p_k(x) \quad on \quad [x_k, x_{k+1}],\ k = 0, ...n. \tag{3.43}$$

Nel caso generale, i pezzi polinomiali $\{p_k(x)\}$ sono costruiti in modo tale che siano tra loro indipendenti, e quindi non dovrebbero neanche costituire una funzione continua $s(x)$ sull'intervallo $[a, b]$. Questo però è un modo inaccettabile di approssimare, particolarmente per funzioni che dovrebbero descrivere un processi di ***levigamento***. In una situazione del genere è naturale richiedere che i pezzi polinomiali $p_k$ si uniscano con fluidità nei punti $x_1, ...., x_n$, cioè, tutte le derivate di $p_{k-1}$ e $p_k$, fino ad un certo ordine, devono coincidere in $x_k$. Come risultato otteniamo una funzione polinomiale fluida definita a tratti , chiamata ***funzione spline***.

**Definizione 3.6.1.** *La funzione $s(x)$ è chiamata funzione spline (o semplicemente "una spline") di grado **r** con nodi $\{x_k\}_1^n$ se $-\infty =: < x_0 < x_1 < ... < x_n < x_{n+1} := +\infty$ e*

1. $\forall k = 0, ..., n \quad s(x)$ *coincide in* $(x_k, x_{k+1})$ *con un polinomio di grado non più grande di **r**;*
2. $s(x), s^x(x), ..., s^{r-1}(x)$ *sono delle funzioni continue sull'intervallo* $(-\inf, +\inf)$.

Ogni polinomio algebrico è una spline senza nodi. Si può vedere dalla definizione 3.6.1 che la $r - esima$ derivata di una spline di grado $r$ con nodi $\{x_k\}_1^n$ è una funzione costante definita a tratti con interruzioni, eventualmente, in $x_1, ..., x_n$. Di contro, l'$r - esima$ funzione primitiva di una funzione costante a tratti è una spline di grado $r$.

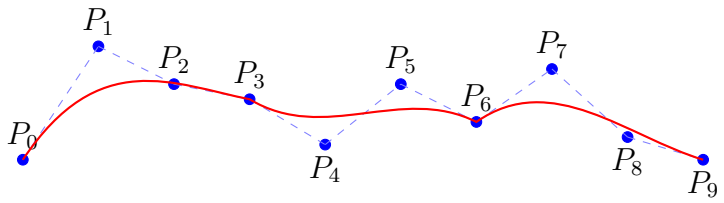



Figura 3.9: Esempio di spline monodimensionale per l'approssimazione dei dati.

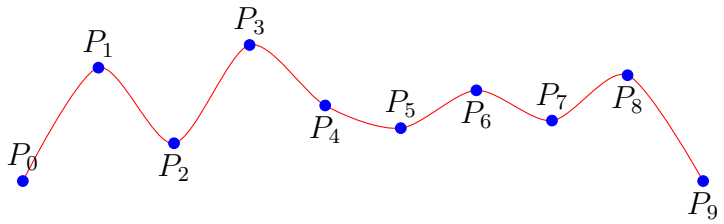



Figura 3.10: Esempio di una spline di interpolazione dei dati.

### 3.6.3 Proprietà delle spline

Le proprietà che andremo ad elencare contribuiscono alla versatilità e all'utilità delle spline in varie applicazioni. Adesso faremo un piccolo accenno ad ognuna di esse:

- **Regolarità**: proprietà cruciale delle spline. Si riferisce alla capacità di una spline di essere liscia e priva di discontinuità o bruschi cambiamenti di direzione. Le spline come già detto, spesso vengono utilizzate per modellare curve in modo fluido e realistico. La regolarità garantisce che la curva sia visivamente piacevole e che le derivate sia coerenti lungo l'intera curva.

- **Interpolazione**: proprietà per cui una spline passa attraverso un insieme specificato di punti di dati. Le spline interpolanti sono utilizzate per rappresentare dati in modo preciso. Garantiscono che la curva passi esattamente attraverso i punti noti, fornendo una stima accurata dei valori intermedi tra i vari punti forniti.

- **Approssimazione dei Dati**: proprietà simile all'interpolazione, ma meno vincolante. Le spline approssimanti cercano di modellare una curva che si avvicini ai dati senza necessariamente passare attraverso tutti i punti. Questa proprietà è preziosa quando i dati possono contenere rumore o incertezze, poiché le spline possono estrarre il modello sottostante senza essere influenzate da singoli punti anomali.

- **Continuità**: altra proprietà cruciale delle spline, si riferisce alla transizione fluida tra i pezzi di una spline, sia in termini di valore che di derivata. Le spline possono essere costruite in modo da garantire la continuità in diverse derivate, garantendo che la curva complessiva sia priva di "scatti" o irregolarità.

Insieme, queste proprietà consentono alle spline di rappresentare curve e funzioni in modo preciso, coerente e visivamente attraente. Queste funzioni possono essere modellate per soddisfare requisiti specifici di regolarità, interpolazione, approssimazione dei dati e continuità, a seconda del contesto dell'applicazione.

**Influenza delle derivate sulle spline.** Il controllo delle derivate prime e seconde è un aspetto fondamentale nella costruzione e nella manipolazione delle spline, poiché influisce direttamente sulla forma e sul comportamento della curva. Le derivate prime e seconde forniscono informazioni sulla pendenza e sulla concavità della curva, rispettivamente. Modificando queste derivate, è possibile ottenere effetti specifici sulla curvatura, sulla fluidità e sulla regolarità della spline. Ecco come il controllo delle derivate prime e seconde influenza la forma delle spline:

1. **Controllo della Pendenza (Derivata Prima)**:
    - Aumento della pendenza. Se si aumentano i valori delle derivate prime in un punto specifico, la curva si alzerà più rapidamente in quel punto, creando un'ascesa rapida.
    - Riduzione della pendenza. Diminuendo i valori delle derivate prime in un punto, la curva si solleverà più lentamente, creando un'ascesa più dolce.
    - Continuità nella pendenza. Garantendo che le derivate prime siano continue in corrispondenza dei nodi, si otterrà una transizione fluida tra i pezzi della curva, evitando "scatti" o irregolarità.
2. **Controllo della Concavità (Derivata Seconda)**:
    - Concavità positiva. Se le derivate seconde sono positive in un intervallo, la spline si curverà verso l'alto (convessità) in quella regione.
    - Concavità negativa. Con derivate seconde negative, la spline si curverà verso il basso (concavità) nell'intervallo.
    - Cambiamenti nella concavità. Modificando le derivate seconde è possibile creare transizioni fluide da concavità positiva a concavità negativa, o viceversa, conferendo alla curva forme più complesse.
3. **Regolarità e fluidità**:
    - Derivate continue. Il controllo delle derivate prime e seconde per garantire continuità tra i pezzi della spline contribuisce a una curva fluida e priva di discontinuità.
    - Regolarità delle derivate. Assicurare che le derivate prime e seconde siano continue e regolari può rendere la curva più "liscia". evitando angoli bruschi o cambiamenti di direzione eccessivamente rapidi.
4. **Modellazione precisa**:

- Adattamento dei dati. Manipolando le derivate prime e seconde, è possibile far si che la spline si adatti in modo preciso ai dati noti, passando attraverso i punti di controllo o approssimandoli.

In sostanza, il controllo delle derivate prime e seconde consente di plasmare la spline in base alle esigenze specifiche dell'applicazione. Questo controllo offre una flessibilità significativa nella creazione di curve con varie forme, regolarità e comportamenti, rendendo le spline uno strumento potente per la modellazione e la rappresentazione di dati complessi.

### 3.6.4 Spline cubica

Fra le funzioni polinomiali a tratti quelle più usate nella pratica, anche perché consentono di ottenere ottimi risultati dal punto di vista grafico, sono le spline cubiche ottenute senza utilizzare senza utilizzare i valori $f^{'}(x)$ e imponendo invece condizioni di continuità delle derivate prima e seconda. Per semplicità d'ora in poi indicheremo $f_i = f(x_i),\ f_i^{'} = f^{'}(x_i)$ e $h_i = x_{i+1} - x_i$.

Una funzione reale $s(x) \in C^2[a,b]$ viene chiamata *spline cubica* ([21]) della $f(x)$ se $s(x)$ coincide con un polinomio $s_i(x)$ di gradi al più 3 in ciascun intervallo $[x_i, x_{i+1}]$ e $s(x_i) = f_i,$ per $i = 0, ..., n$.

Imponendo queste condizioni si ottengono $4n - 2relazioni$

**a)** $s_i(x_i) = f_i, \quad s_i(x_{i+1}) = f_{i+1}, \quad$ per $i = 0, ..., n-1$.

**b)** $s_{i-1}^{'}(x_i) = s_i^{'}(x_i), \quad$ per $\quad i = 1, ..., n-1$.

**c)** $s_{i-1}^{''}(x_i) = s_i^{''}(x_i), \quad$ per $\quad i = 1, ..., n-1$.

Poiché i coefficienti dei polinomi $s_i(x)$ sono $4n$, occorre imporre due condizioni aggiuntive al bordo, che vengono scelte caso per caso. Le due più usate sono:

**d')** $s_0^{'}(x_0) = s_{n-1}^{''}(x_n) = 0$ (*spline naturale*), che impone alla spline un andamento lineare vicino agli estremi;

oppure, se sono noti i valori di $f^{'}(a)$ e $f^{'}(b)$

**d")** $s_0^{'}(x_0) = f^{'}(a), \quad s_{n-1}^{'}(x_n) = f^{'}(b)$ (*spline completa*), che impone alla spline la tangenza alla $f(x)$ negli estremi;

Se invece, i valori $f^{'}(a)$ e $f^{'}(b)$ non fossero disponibili, si potrebbero sostituire con delle approssimazioni. Ad esempio, $f^{'}(a)$ potrebbe essere approssimato con la derivata in $a$ del polinomio che interpola la $f(x)$ su $a$ e sui tre nodi successivi.

Sulla derivata seconda si potrebbero imporre anche altre condizioni, per esempio $s_0^{''}(x_0) = \sigma_0$ e $s_{n-1}^{''}(x_n) = \sigma_n$, dove $\sigma_0$ e $\sigma_n$ sono valori assegnati, oppure $s_0^{''}(x_0) = s_0^{''}(x_1)$ e $s_{n-1}^{''}(x_{n-1}) = s_{n-1}^{''}(x_n)$, assumendo che la derivata seconda sia costante vicino agli estremi dell'intervallo $[a,b]$.

**Calcolo dei coefficienti.** Per determinare i coefficienti dei polinomi $s_i(x)$ si potrebbero sfruttare direttamente le condizioni a) - c) e le condizioni aggiuntive scelte, risolvendo un sistema lineare di $4n$ equazioni in $4n$ incognite. E' possibile però semplificare il problema riducendo il numeri delle equazioni necessarie. Si considerino come incognite le quantità, dette **momenti**,

$$\mu_i = s_i^{''}(x_i), \quad \text{per } i = 0, ..., n-1, \quad \text{e } \mu_n = s_{n-1}^{''}(x_n). \tag{3.44}$$

Poiché $s_i(x)$ per $x \in [x_i, x_{i+1}]$ è un polinomio di grado al più 3, $s_i^{''}$ è il polinomio di grado al più 1

$$s_i^{''}(x) = u_{i+1}\frac{x - x_i}{h_i} - u_i\frac{x - x_{i+1}}{h_i}. \tag{3.45}$$

Integrando due volte si ottiene

$$s_i^{'}(x) = u_{i+1}\frac{(x - x_i)^2}{2h_i} - u_i\frac{(x - x_{i+1})^2}{2h_i} + \alpha_i. \tag{3.46}$$

$$s_i(x) = u_{i+1}\frac{(x - x_i)^3}{6h_i} - u_i\frac{(x - x_{i+1})^3}{6h_i} + \alpha_i(x - x_i) + \beta_i. \tag{3.47}$$

Le costanti $\alpha_i$ e $\beta_i$ vengono determinare imponendo le condizioni a) (per maggiori dettagli sui vari passaggi si rimanda alla bibliografia [21])

$$\begin{cases} \beta_i = f_i - \mu_i\frac{h_i^2}{6} \\ \alpha_i = \frac{f_{i+1} - f_i}{h_i} - \frac{h_i}{6}(\mu_{i+1} - \mu_i). \end{cases} \tag{3.48}$$

Restano da calcolare i $\mu_i$ per $i = 0, ..., n$. Detto che

$$f_{i-1,i,i+1} = \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}} \tag{3.49}$$

i $\mu_i$ sono le soluzioni di un sistema lineare, ottenuto associando diverse equazioni a seconda che debbano essere verificate le condizioni d') o d").

Nel primo caso, tenendo conto che $\mu_0 = \mu_n = 0$, il sistema che si ottiene è

$$M \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ . \\ \mu_{n-2} \\ \mu_{n-1} \end{bmatrix} = 6 \begin{bmatrix} f_{0,1,2} \\ f_{1,2,3} \\ . \\ . \\ . \\ f_{n-3,n-2,n-1} \\ f_{n-2,n-1,n} \end{bmatrix}, \tag{3.50}$$

Nel secondo caso vengono aggiunte una prima e un'ultima equazione al sistema che diventa

$$M \begin{bmatrix} \mu_0 \\ \mu_1 \\ . \\ . \\ . \\ \mu_{n-1} \\ \mu_n \end{bmatrix} = 6 \begin{bmatrix} f_{0,0,1} \\ f_{0,1,2} \\ . \\ . \\ . \\ f_{n-2,n-1,n} \\ f_{n-1,n,n} \end{bmatrix}, \tag{3.51}$$

La matrici $M$ in (3.50) e (3.51) sono a predominanza diagonale e quindi sono non singolari (cioè $det(M) \neq 0$). Perciò i sistemi hanno una e una sola soluzione, che può essere calcolata con il metodo di Gauss senza adoperare nessuno scambio di righe. Inoltre le matrici tridiagonali e il metodo di Gauss ha un basso costo computazionale, dell'ordine di $n$. Sempre per maggiori dettagli sull'intera operazione di calcolo dei coefficienti di una spline si rimanda al materiale indicato in bibliografia [21], anche per alcuni esempi di spline naturale e completa.

**Costo computazionale.** Il costo computazionale viene determinato a meno di termini costanti rispetto a $n$. Indicando con $A$ le operazioni additive e con $M$ le operazioni moltiplicative, per la costruzione dei coefficienti e dei termini noti del sistema lineare (3.50) sono richieste $4nA$ e $3nM$. Per la risoluzione del sistema tridiagonale sono richieste $3nA$ e $5nM$. Per il calcolo degli $\alpha_i$ e $\beta_i$ sono richieste $3nA$ e $3nM$. Quindi in totale la costruzione della spline richiede $10nA$ e $11nM$.

Per calcolare il valore della spline in un punto $x$ è necessario prima individuare l'indice $i$ tale che $x \in (x_i, x_{i+1})$. Per ottenere $i$ si può usare l'algoritmo banale (cioè confrontare $x$ successivamente con $x_j$, $j = 1, \ldots, n-1$) e questo richiede $n-1$ confronti. Se però $n$ è potenza di 2, si può usare l'algoritmo di bisezione (cioè confrontare $x$ con $x_{n/2}$, se $x$ è minore di $x_{n/2}$ si confronta $x$ con $x_{n/4}$, se $x$ è maggiore di $x_{n/2}$ si confronta $x$ con $x_{3n/4}$, e così via). Questo procedimento richiede $\log_2 n$ confronti. Una volta trovato $i$, per calcolare $s(x)$ sono richieste $5A$ e $9M$.

## 3.7 Sincronizzazione di due fonti video

La sincronizzazione di due fonti video è un problema complesso nell'ambito dell'elaborazione del segnale e dell'ingegneria multimediale. Questo processo richiede la comprensione accurata delle proprietà temporali delle fonti video coinvolte, inclusi i timestamp dei frame, i ritardi di trasmissione e le possibili fluttuazioni nella velocità di riproduzione. Gli algoritmi di sincronizzazione devono affrontare sfide come la variazione di latenza nei dispositivi di cattura e di riproduzione, nonché i ritardi introdotti dai processi di compressione e trasmissione.

Una delle tecniche comuni per affrontare il problema della sincronizzazione è l'uso di metodi di cross-correlazione tra i segnali video. Questi metodi cercano pattern simili tra le fonti video e utilizzano l'informazione di sincronizzazione per allineare i frame in base a massimi o minimi di correlazione. Altre strategie coinvolgono l'analisi delle caratteristiche visive e audio dei video per stimare i tempi di ritardo e correggere gli scostamenti temporali.

La precisione della sincronizzazione è essenziale in applicazioni come la realtà virtuale e aumentata, dove anche un piccolo errore temporale può compromettere l'esperienza utente. Le tecnologie di sincronizzazione avanzate sono utilizzate per garantire che i contenuti visivi e audio siano allineati con le azioni dell'utente in tempo reale, creando una sensazione di coerenza e immersività.

Nell'ambito delle trasmissioni in diretta, la sincronizzazione video è cruciale per evitare disallineamenti tra video e audio, che possono essere sgradevoli per gli spettatori. Gli algoritmi di compensazione del ritardo vengono spesso utilizzati per allineare l'audio e il video in modo da garantire un'esperienza di visione sincronizzata.

Inoltre, la sincronizzazione video ha implicazioni nell'ambito della sicurezza e della videosorveglianza, dove è importante avere un'accurata sequenza temporale dei frame per analisi forensi e rilevamento di eventi critici.

In conclusione, la sincronizzazione di due fonti video è un problema tecnico di rilievo che richiede un approccio scientifico e ingegneristico per affrontare le sfide legate alle proprietà temporali dei segnali video. Gli algoritmi e le tecniche utilizzate per la sincronizzazione hanno un impatto significativo su diversi settori, influenzando l'esperienza utente, l'efficacia delle applicazioni interattive e la qualità delle trasmissioni video.

### 3.7.1 SMPTE Timecode

Lo SMPTE timecode (alcuni riferimento per approfondire l'argomento [22], [23], [24], [25], [26]), acronimo di "Society of Motion Picture and Television Engineers timecode," è un sistema di codifica temporale ampiamente utilizzato nell'industria cinematografica, televisiva e multimediale per sincronizzare in modo preciso diverse fonti audio e video. Questo codice temporale è costituito da una sequenza numerica di cifre che rappresenta le ore, i minuti, i secondi e i frame di un video. Ogni frame riceve un numero univoco, consentendo un riferimento temporale esatto all'interno di un progetto multimediale. SMPTE timecode è presentato nel formato ore:minuti:secondi:frame ed è tipicamente rappresentato in codice binario a 32 bit.

L'utilizzo dello SMPTE timecode è cruciale in scenari in cui è necessario allineare o sincronizzare diverse fonti video o audio. Ad esempio, durante la produzione cinematografica, il timecode viene impiegato per coordinare l'audio e il video provenienti da diverse telecamere o registrazioni audio, semplificando il processo di montaggio e post-produzione.

Nelle trasmissioni in diretta, lo SMPTE timecode è essenziale per garantire una sincronizzazione accurata tra gli input video e audio provenienti da varie fonti. Questo sistema di codifica temporale facilita anche l'identificazione e la localizzazione di specifici momenti o clip all'interno di un progetto, semplificando il flusso di lavoro nella post-produzione e agevolando la collaborazione tra diversi professionisti dell'industria multimediale.

Inoltre, lo SMPTE timecode può essere utilizzato in applicazioni interattive, come simulatori e giochi, per garantire la sincronizzazione precisa tra le azioni dell'utente e gli eventi nel mondo virtuale. Questo contribuisce a creare un'esperienza utente più coinvolgente e realistica.

In sintesi, lo SMPTE timecode è uno strumento fondamentale nella sincronizzazione video e svolge un ruolo chiave nell'assicurare che diverse fonti audio e video siano allineate con precisione temporale. Questo sistema di codifica è ampiamente adottato nell'industria multimediale per garantire una produzione di alta qualità e un'esperienza di visione ottimale per gli spettatori.

# Capitolo 4

# Tecnologia utilizzate

In questo capitolo discuteremo le tecnologie ed i software utilizzati nello sviluppo del progetto. In particolare per ciascuna tecnologia metteremo in evidenza i seguenti aspetti:

- utilizzo.
- Responsabile della scelta e motivazioni.
- Funzionalità e vantaggi.
- Conoscenza a priori.

## 4.1 YOLOv5

Presentiamo questa tecnologia perché anche se questo strumento non verrà usato in prima persona dal sistema implementato in questo lavoro di tesi, è una parte fondamentale dello stesso che produce una serie di risultati che sono gli input delle procedure sviluppate.

Gli algoritmi di Intelligenza Artificiale e di elaborazione delle immagini faranno uso di funzionalità largamente utilizzate nel dominio della visione artificiale: classificazione di oggetti, riconoscimento del movimento e tracciamento di elementi nella scena ripresa. A titolo di esempio nelle figure 4.1 e 4.2 si mostra il risultato di **Object Detection** ottenuto con il software open source YOLOv5 (You Look Only Once v5) [7].

Come già scritto, questa tecnologia non è impiegata in prima persona per le parti di progetto descritte in questa tesi. Inoltre, è stato un primo approccio a questo tipo di tecnologie, però è risultato molto interessante capire come funzionassero, almeno in parte, i sistemi di riconoscimento di oggetti. I vantaggi offerti da questa tecnologia sono stati spiegati in parte prima, ma si può benissimo capire come avere a disposizione uno strumento del genere abbia alleggerito tantissimo il lavoro di analisi dell'immagine che si sarebbe dovuto fare sui video per ottenere la posizione della moto e quella di ogni cono, per ogni frame, visto che il riconoscimento è fatto completamente da una rete neurale, previa allenamento della stessa. Questo non vuol dire che lo


Figura 4.1: a)


Figura 4.2: b)

Figura 4.3: a) Riconoscimento automatico di motocicli, e b) coni segnaletici mediante il software open source YOLOv5. L'indicazione della classe di oggetto riconosciuto è sempre collegata ad una valore di confidenza di tale risultato.

strumento sia in grado di fare tutto da solo, ma è stato necessario formattare in determinati modi i dati che venivano restituiti.

## 4.2 OpenCV

**Computer Vision.** Come umani, percepiamo la struttura tridimensionale del mondo attorno a noi con una apparente facilità. Possiamo distinguere le forme e la trasparenza di ogni oggetto che ci troviamo davanti, la luce e le ombre che si generano attraverso le superfici. Guardando ad una foto di gruppo, possiamo facilmente contare e riconoscere tutte le persone nella foto ed anche capire quali sono le loro emozioni dalle espressioni facciali (Figura 4.4a). Gli psicologi percettivi hanno passato decenni a cercare di capire come funziona il sistema visivo umano e, anche se possono ideare alcune illusioni ottiche per prendere in giro alcuni dei suoi principi, una completa soluzione a questo puzzle rimane ancora un mistero.

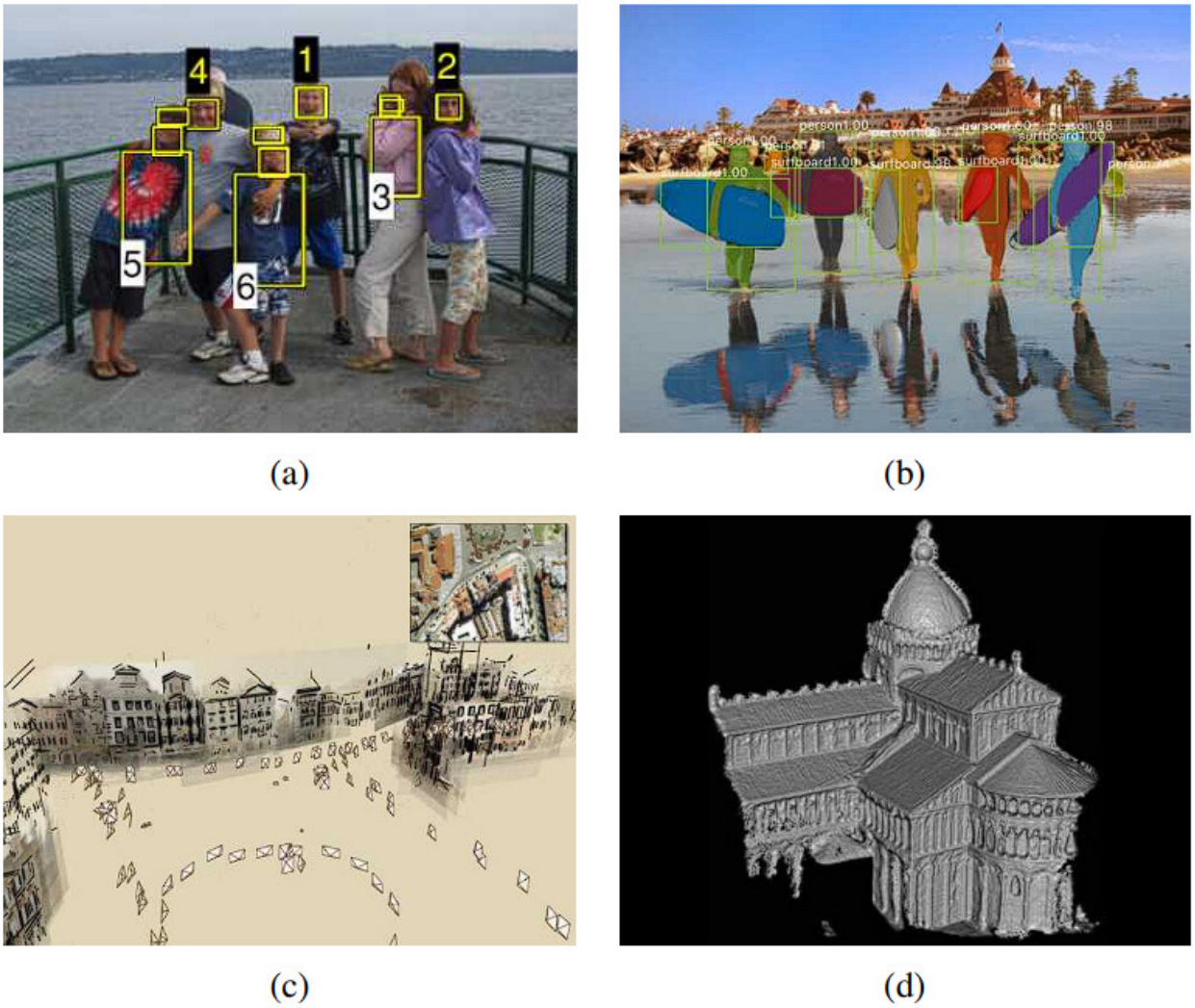



Figura 4.4: Alcuni esempi di algoritmi di Computer Vision e sue applicazioni

I ricercatori nel campo della Computer Vision ([27], [28], [29]) hanno sviluppato, in parallelo tecniche matematiche per ricostruire la forma tridimensionale e l'aspetto degli oggetti nelle immagini. Il progresso nelle ultime due decadi è stato molto rapido. Ad oggi esistono tecniche

affidabili per calcolare con precisione modelli 3D di un ambiente a partire da migliaia di fotografie parzialmente sovrapposte (Figura 4.4c). Dato un insieme sufficientemente ampio di vedute di un particolare oggetto o facciata, possiamo creare modelli di superficie 3D densi e accurati (Figura 4.4d).

Tuttavia, nonostante tutti questi progressi, il sogno di avere un computer che spiega una immagine con lo stesso livello di dettaglio e causalità di un bambino di due anni rimane lontana. Sorge spontanea una domanda. Perché la Computer Vision è così complicata? In parte, è perché questo è un problema inverso, nel qualche cerchiamo di recuperare alcune informazioni sconosciute e insufficienti per specificare completamente la soluzione. E' necessario quindi ricorrere a *modelli* fisici e probabilistici, oppure al *machine learning* a partire da un grande insieme di esempi, per disambiuguizzare le potenziali soluzioni. Tuttavia, modellare quello che vede un essere umano in tutta la sua ricca complessità è molto più difficile di quello che si possa pensare.

I modelli *forward* che si usano nella Computer Vision sono solitamente sviluppati in fisica e nella Computer Grafica. Entrambi i campi modellano come muovere e animare gli oggetti, come la luce riflette sulle superfici, si sparpaglia nell'atmosfera, rifratta attraverso le lenti di una telecamera (o l'occhio umano), ed infine proiettata sul piano di una immagine piatta. Sebbene la computer grafica non sia ancora perfetta, in molti campi, come il rendering di una scena fissa composta da oggetti di uso quotidiano oppure animare creature esistenti, l'illusione della realtà è presente.

Nella Computer Vision, cerchiamo di fare il contrario, cioè, cerchiamo di descrivere il mondo che vediamo in una o più immagini ricostruendone le sue proprietà, come la forma, l'illuminazione e la distribuzione di colore. E' sorprendente come gli esseri umani e gli animali lo facciano senza sforzo, mentre gli algoritmi di Computer Vision sono così soggetti a errori. Ad oggi la Computer Vision è usata in una grande varietà di applicazione del mondo reale, i quali sono:

- **Riconoscimento ottico dei caratteri (OCR)**: leggeri codici postali scritti a mano sulle lettere (Figura 4.5a), riconoscimento automatico delle targhe di veicoli ([30], [31]).
- **Ispezione macchinari**: rapida ispezione delle parti per controlli di qualità mediante visione stereoscopica con illuminazione specializzata per misurare le tolleranze sulle ali degli aerei o parti di carrozzeria (Figura 4.5b).
- **Vendita al dettaglio**: riconoscimento di oggetti per corsie di cassa automatizzate e negozi completamente automatizzati ([32]).
- **Logistica di magazzino**: consegna autonoma dei pacchi e prelievo di parti tramite braccio robotico (Figura 4.5c, [33], [34]).
- **Immagini mediche**: registrazione delle immagini preoperatorie e intraoperatorie (Figura 4.5d), o esecuzione a lungo termine di studi sulla morfologia del cervello umano durante l'invecchiamento.
- **Veicoli a guida autonoma**: capaci di guidare punto a punto tra città (figura 4.5e, [35]).

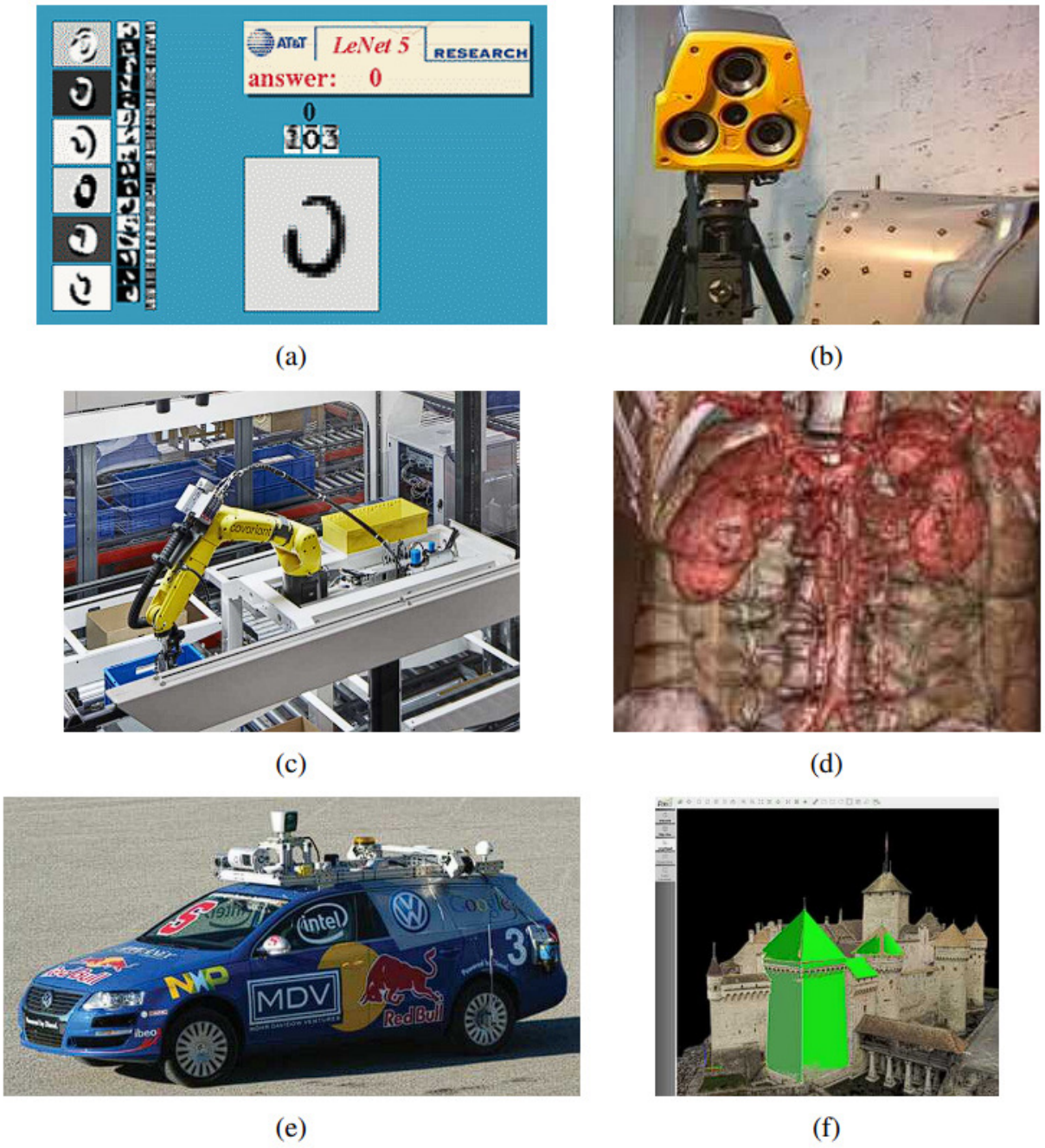



Figura 4.5: Alcune applicazioni industriali della Computer Vision

- **Modelli 3D di edifici**: costruzione completamente automatizzata di modelli 3D da fotografie aeree e da droni (figura 4.5f) https://www.pix4d.com/.
- **Match Move**: unire le immagini generate in computer grafica (CGI) con riprese dal vivo tracciando punti caratteristici nel video sorgente per stimare il movimento e la forma della telecamera 3D nell'ambiente ([36]).
- **Motion capture**: utilizzando dei marcatori retroriflettenti visualizzati da più telecamere o altre tecniche basate sulla visione per catturare attori per l'animazione al computer.
- **Riconoscimento impronte e biometrica**: per accessione via autenticazione automatico, come anche applicazioni legali.

Anche se le applicazioni sopra sono tutte estremamente importanti, si riferiscono per lo più a tipi di immagini specializzate e domini abbastanza ristretti. In aggiunta a tutte queste applicazioni industriali, esistono una miriade di applicazione a livello consumatore:

- **Stitching**: trasformando le foto sovrapposte in un unico panorama cucito senza soluzione di continuità (figura 4.6a).
- **Supporto all'esposizione**: fondere diverse esposizioni acquisite sotto condizioni di illuminazioni impegnative (luci troppo forti e ombre) in una singola immagine con una perfetta esposizione (figura 4.6b).
- **Trasformazione**: modificando la foto di una persona in un'altra, usando una metamorfosi senza cuciture (figura 4.6c).
- **Modellazione 3D**: convertire una o più istantanee in un modello 3D dell'oggetto o persona che si sta fotografando (figura 4.6d).
- **Navigazione attraverso immagini**.
- **Riconoscimento facciale**.
- **Autenticazione visiva**.

La maggior parte di quello di cui si occupa la Computer Vision riguarda il trovare soluzioni di problemi inversi o la stima di quantità non note, e per trovare queste soluzioni sono necessari degli **algoritmi**. Per molti problemi di visione, è abbastanza semplice trovare una descrizione matematica del problema che non corrisponda a quella realistica condizione nel mondo relate o non si presta alla stima stabile delle incognite. Quello di cui abbiamo bisogno sono algoritmi che sono al tempo stesso **robusti** rispetto al rumore e deviazioni dal modello e ragionevolmente **efficienti** in termini di tempo di esecuzione e spazio utilizzato.

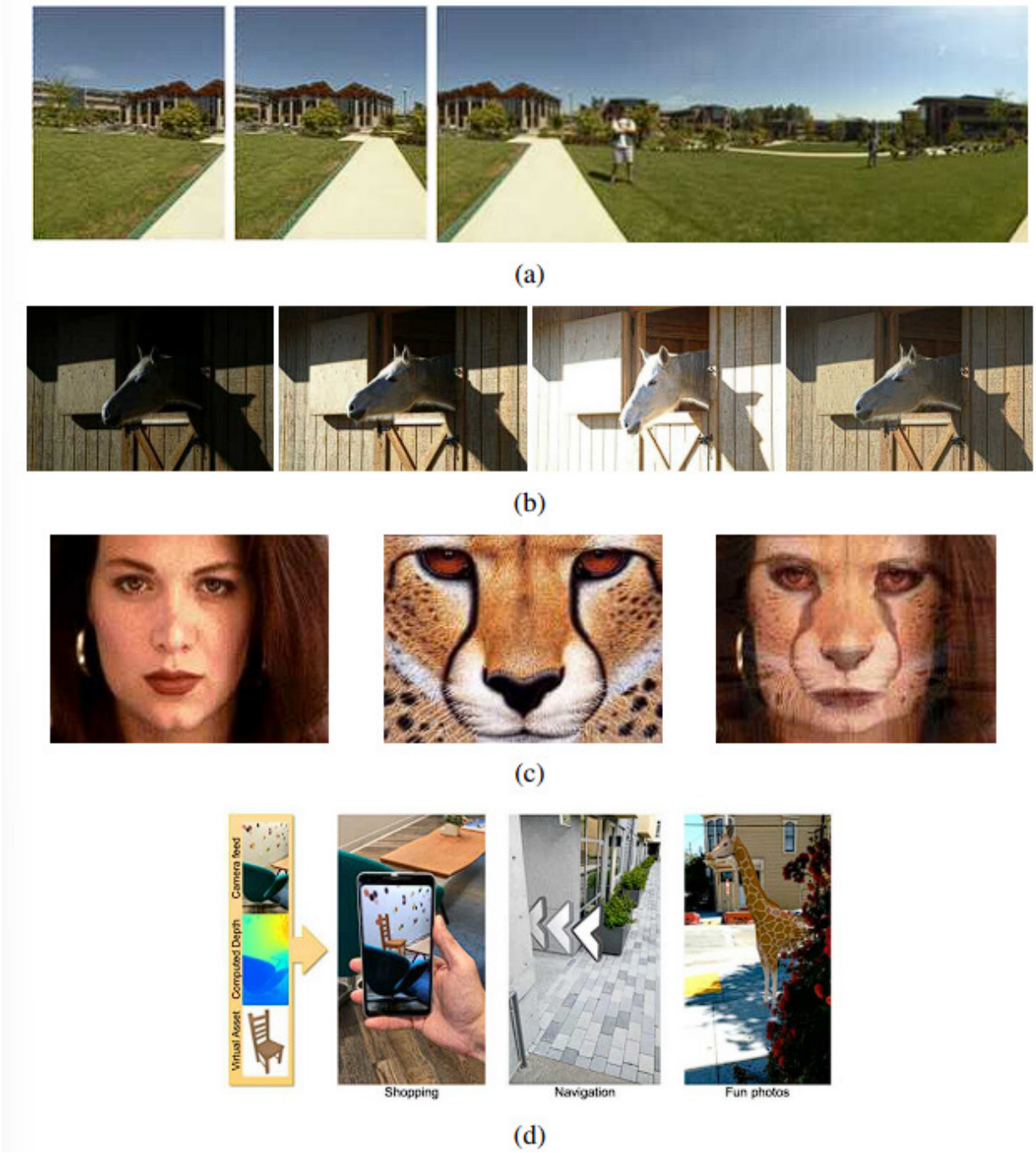



Figura 4.6: Alcune applicazione a livello consumatore della Computer Vision (pagina 9-10 libro [27]).

**OpenCV.** OpenCV (Open Source Computer Vision Library [37]) è una libreria software open source di computer vision e machine learning. OpenCv è stata creata per fornire una infrastruttura comune per le applicazioni di Computer Vision e per accelerare l'uso della percezione di macchina nei prodotti commerciali. Essendo un prodotto con licenza Apache2, OpenCV semplifica l'utilizzo e la modfica del codice da parte di chiunque ne abbia bisogno.

La libreria a più di 2500 algoritmi ottimizzati, che include un set completo di algoritmi di computer vision e machine learning classici e all'avanguardia. Questi algoritmi possono essere usati per riconoscere facce, identificare oggetti, classificare azione umani in video, tracciare i movimenti di una videocamera, tracciare il movimento di oggetti, estrarre modelli 3D da oggetti, produrre nuvole di punti 3D da camere stereo, unire immagini insieme per produrre un'unica immagine ad alta risoluzione di un'intera scena, trovare immagini simili da un database di immagini, rimuovere gli occhi rossi da una foto scattata con flash, seguire il movimento degli occhi, riconoscere lo scenario e stabilire marcatori per sovrapporlo con la realtà aumentata, etc. OpenCV ha più di 47mila persone nella comunità di utenti e un numero stimato di download superiore a 18 milioni. La libreria è ampiamente utilizzata in varie aziende, gruppi di ricerca ed enti governativi.

Insieme ad aziende affermate come Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota che utilizzano la libreria, ci sono molte startup come Applied Minds, VideoSurf e Zeitera, che fanno ampio uso di OpenCV. Gli usi implementati di OpenCV spaziano dall'unione delle immagini di StreetView, alla rilevazione di intrusioni in sistemi di sorvegliaza a Israele, monitoraggio delle attrazzature minerarie in Cina, aiutare i robota a muoversi e raccogliere oggetti al Willow Garage, rilevamento di incidenti di annegamento in piscina in Europa, gestire arte interattiva in Spagna e New York, ispezione delle etichette sui prodotti nelle fabbriche di tutto il mondo al rilevamento dei volti in Giappone.

Ha interfacce in C++, Python, Java e MATLAB e supporta Windows, Linux, Android e Mac OS. OpenCV si appoggia principalmente alle applicazioni in tempo reale e sfrutta le istruzioni MMX e SSE quando disponibili. Ci sono oltre 500 algoritmi e circa 10 volte tante funzioni che compongono e supportano tali algoritmi. OpenCv è scritto nativamente in C++ e dispone di un'interfaccia basata su modelli che funzionano perfettamente con contenitori STL.

Anche per questa libreria è stato un primo approccio, ma grazie al team siamo stati messi in grado di comprenderne il funzionamento molto velocemente. Il motivo per cui è stata scelta riguarda sicuramente il fatto che il team che ha lavorato al progetto usa da tantissimi anni questa libreria ed inoltre è conosciuta a livello mondiale nel campo della Computer Vision.

## 4.3 Unreal Engine 5

Unreal Engine 5 (UE5 [38]) rappresenta un punto di svolta nell'industria dei motori grafici, offrendo una serie di funzionalità avanzate e vantaggi significativi per gli sviluppatori di giochi e professionisti del settore.

Nel panorama degli strumenti di sviluppo di videogiochi, Unreal Engine 5 si distingue come uno dei motori grafici più potenti e versatili disponibili. Progettano da Epic Games, UE5 è la succesiva iterazione nella serie di motori grafici Unreal Engine, mirando a superare i limiti della tecnologia e a fornire una piattaforma all'avanguardia per la creazione di esperienze interattive immersive.

UE5 trova applicazione in una vasta gamma di settori, non solo nel mondo dei videogiochi ma anche nella produzione cinematografica, nell'architettura, nella simulazione e nell'addestramento virtuale. Questo motore offre una solida base per sviluppare giochi coinvolgenti, esperienze virtuali realistiche e progetti di visualizzazione avanzata.

Quando si tratta di scegliere un motore grafico, UE5 offre vantaggi significativi rispetto ad altre opzioni disponibili sul mercato. Questi vantaggi includono:

- **Nanite Virtualized Geometry**: Una delle funzionalità più impressionanti di UE5 è la tecnologia Nanite, che consente di gestire dettagli geometrici incredibilmente complessi senza sacrificare le prestazioni. Ciò consente di creare mondi e ambienti dettagliati con una minore preoccupazione per l'ottimizzazione estrema.

- **Lumen Global Illumination**: Grazie a Lumen, il sistema di illuminazione globale in tempo reale di UE5, gli sviluppatori possono creare ambienti realistici con riflessi e ombre dinamiche. Questo contribuisce a un'esperienza visiva più coinvolgente e convincente.

- **Workflow creativo e Strumenti integrati**: UE5 offre un'ampia gamma di strumenti integrati per semplificare il processo di sviluppo. Il sistema di animazione, la fisica avanzata e gli strumenti di livello sono progettati per consentire agli sviluppatori di esprimere la propria creatività senza limitazioni.

- **Community e supporto**: L'ecosistema di Unreal Engine comprende una vasta comunità di sviluppatori, artisti e professionisti del settore. Cià significa che ci sono risorse, tutorial e supporto facilmente accessibili per risolvere problemi e migliorare le competenze nell'utilizzo di questo motore.

In sintesi, UE5 rappresenta un passo avanti nell'evoluzione dei motori grafici, offrendo funzionalità all'avanguardia che consentono agli sviluppatori di creare esperienze di gioco e progetti di visualizzazione di alta qualità. La sua combinazione di potenza, flessibilità e strumenti creativi lo rende una scelta attraente per coloro che cercano di portare la propria visione in vita attraverso il mondo digitale.

La scelta di utilizzare questo motore grafico è stata a cura dell'autore di questa tesi. Si avevano due possibili scelte, Unity o UnrealEngine, ma alla fine si è optato per la seconda. Si sente sempre

parlare di motori grafici per videogiochi, ma non c'era mai stata la possibilità di poterne utilizzare uno. Il motivo per cui è stato scelto UE5 riguarda principalmente il linguaggio che utilizza il software, cioè C++, ed il fatto che c'è dietro ci sia una comunità così ampia che è possibile trovare in rete qualsiasi cosa di cui si abbia bisogno.

# Capitolo 5

# Lavoro svolto

Il lavoro svolto in questo tirocinio è ben rappresentato dalla parte colorata nella figura 5.1, che descrive il flusso di elaborazione dell'intero sistema. I rettangoli viola indicano le funzionalità di elaborazione dell'immagine e quelli rossi le segnalazioni di eventuali penalità ricevute. Nelle sezioni successive si descriverà nel dettaglio tutto il lavoro effettuato in ognuna delle fasi indicate e verranno anche illustrate le idee di base dietro la segnalazione di ogni penalità connessa al test di guida, insieme ai metodi utilizzati per mettere in risalto queste penalità. In questa sezione andremo a parlare solamente delle fasi oggetto del lavoro svolto per questo tirocinio; se si volesse sapere di più sul conto delle fasi precedenti a quella di "Proiezioni" si faccia riferimento al capitolo 2.

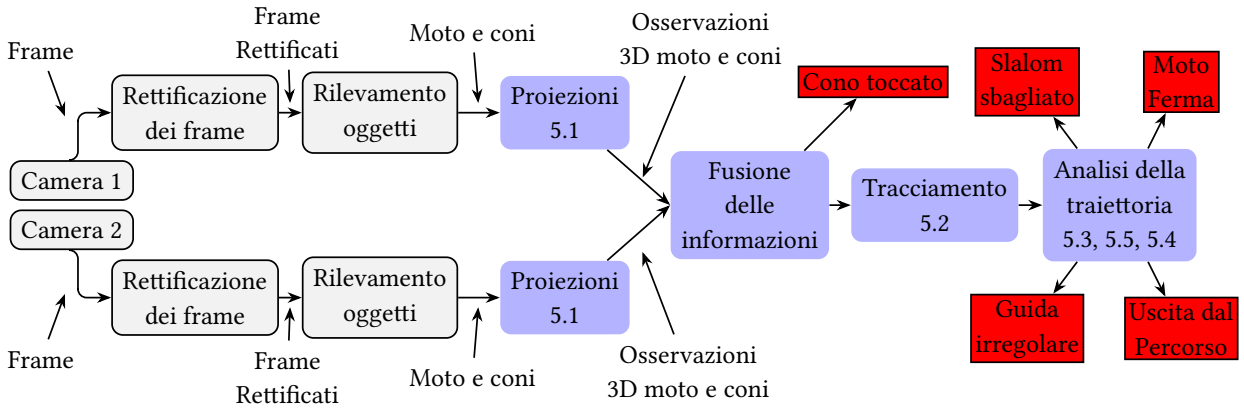



Figura 5.1: Architettura completa del sistema: le parti colorate rappresentano le funzionalità implementate in questo progetto di tesi, mentre quelle bianche verranno assunte come già implementate ma pur sempre illustrate nel loro funzionamento (vedere Cap. 2). I rettangoli viola senza bordo indicano le funzionalità di elaborazione delle immagini, mentre quelli rossi le segnalazioni di eventuali penalità rilevate.

Si noti come nella figura 5.1 a differenza della figura 2.3, viene specificato il fatto che il numero delle telecamere utilizzato fosse pari a due. Questo scelta è motivata dal fatto che nel sistema installato per monitorare la pista LSB , il numero di telecamere utilizzato è proprio due, una

che riprende la pista verticalmente (ci riferiremo ad essa come L1 da ora in poi) mentre l'altra orizzontalmente (ci riferiremo ad essa come L2 ). L'architettura generale del sistema è basata su un numero imprecisato di telecamere che riprendono la pista (LSB o HSA che sia), infatti la pista HSA non è ripresa da due sole telecamere ma necessariamente da più di due. Fa parte del lavoro di questo tirocinio unire le varie informazioni ottenute dalle N telecamere montate sulla pista per ottenere il miglior insieme di informazioni possibile. Durante il lavoro svolto per questo tirocinio ci siamo occupati solamente della pista LSB, quindi tutti i risultati e immagini che verranno introdotti in questa sezione faranno riferimento a quella pista.

**Dimensioni pista di riferimento.** Prima di analizzare nel dettaglio le fasi successive a quella di Rilevamento degli oggetti, vogliamo prima comprendere e delineare bene la situazione in cui ci troviamo, per cercare di capire se quello che vogliamo fare sia effettivamente possibile. Prima di tutto cerchiami di definire alcune questioni chiave:

- c'è un solo oggetto mobile che ci interessa tracciare, e si tratta della moto.
- Si consideri la traccia ripresa da solo una delle due telecamere. La fusione delle informazioni relative a due fonti video verrà trattata più avanti.
- Conosciamo già le dimensione della pista in esame. Grazie a questo è possibile fare un'equivalenza abbastanza precisa pixel/cm per ogni video. In questo ci aiutano anche i coni , perché conosciamo la distanza che ci sono fra loro (come si può vedere in figura 2.1).

Riportiamo le dimensioni della pista HSA (verrà spiegato in seguito il perché ci stiamo concentrando su questa pista, nonostante la pista di riferimento sia la LSB).

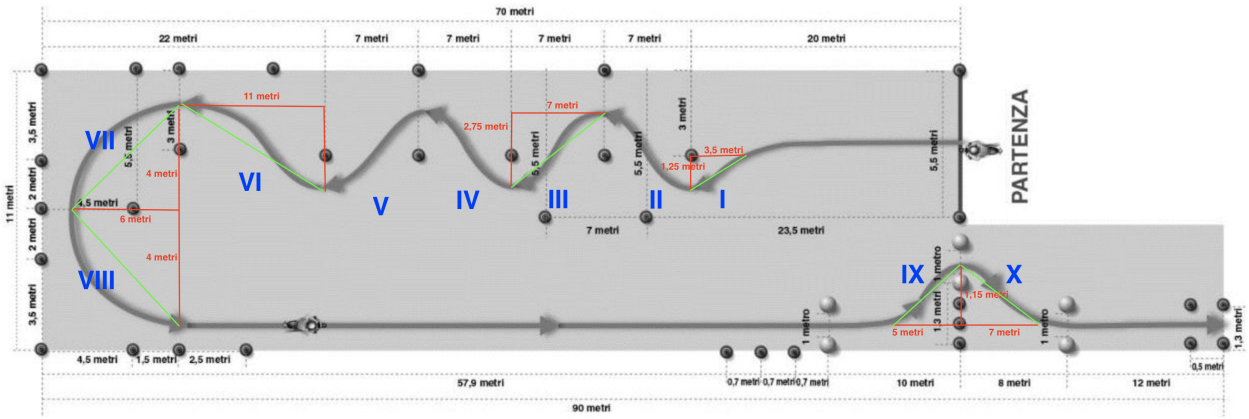



Figura 5.2: Dimensioni pista HSA

Osservando l'immagine 5.2, se per comodità poniamo l'origine del piano cartesiano nel cono in basso a sinistra, allora abbiamo che le dimensioni dello scenario sono le seguenti:

$$11m \cdot 90m = 1100cm \cdot 9000cm \tag{5.1}$$

si considerino le dimensioni di un cono (20cm - 30cm) e lo spostamento apprezzabile della moto di $1cm$, allora la distanza (approssimata) che la moto percorrerà sarà la seguente:

$$I = \sqrt{3,5^2 + 1,5^2} = 3,80m \tag{5.2}$$

$$II = \sqrt{7^2 + 2,75^2} = 7,52m \tag{5.3}$$

$$III = IV = V = II \tag{5.4}$$

$$VI = \sqrt{11^2 + 2,75^2} = 11,33m \tag{5.5}$$

$$VII = \sqrt{4^2 + 6^2} = 6,32m \tag{5.6}$$

$$VIII = VII \tag{5.7}$$

$$IX = \sqrt{5^2 + 1,15^2} = 5,13m \tag{5.8}$$

$$X = \sqrt{7^2 + 1,15^2} = 7,09m \tag{5.9}$$

$$L_{percorso} = 20m + I - 3,5m - 6m + II + III + IV + V + VI + VII + VIII + \tag{5.10}$$

$$+60m + 5m + IX + X + 1m + 12m = 158,57m \tag{5.11}$$

Considerando anche lo spazio necessario per percorrere i vari slalom e fare manovre di altro genere, la distanza che la moto percorrerà sarà all'incirca di $160m$.

Il decreto ministeriale qui citato [5] impone che la pista HSA sia completata entro 25sec. Pensando al caso pessimo per la nostra situazione, supponiamo anche che la durata media di un percorso sia di $20sec$, il quale è un tempo che fanno veramente in pochi secondo quanto ci è stato detto da un istruttore di guida. Allora la velocità media della moto, secondo quanto visto nella sezione 3.2.1 sarà data da $spazio/tempo$. Si osserva che questa velocità sarà inferiore nella parte dello slalom e superiore nel rettilineo di ritorno.

$$v_m \approx \frac{spazio}{tempo} = \frac{160\ m}{20\ s} \approx 8\ \frac{m}{s} \tag{5.12}$$

**Caratteristiche della telecamera a nostra disposizione.** Le nostre due telecamere installate sul posto, registrano in 1080p a 50fps. Riferendoci a quanto scritto nella sezione 3.3 si può affermare che la **frequenza di campionamento** $f_c$ è:

$$f_c = \frac{1}{50}\ sec \tag{5.13}$$

che espressa in Hertz sarebbe $50Hz$, cioè passa $1/50sec$ tra un frame ed il successivo.
Secondo quanto scritto nel teorema 3.3.1, raggiungiamo la conclusione che con le due telecamere possiamo campionare bene movimenti che avvengono in un tempo che sia almeno maggiore di 1/25sec, oppure se vogliamo dirla in termini di frequenze riusciamo a campionare bene un segnale che varia al massimo con $f_c = 25Hz$.
Cercando di esplicare meglio quanto appena scritto, possiamo distinguere in due casi:

1. Se un cono appena toccato oppure la moto stessa si spostassero in un tempo $t \leq \frac{1}{25}sec$ non sarei più capace di ricostruire il movimento di quell'oggetto, che si è mosso troppo velocemente.

2. Di contro maggiore è il tempo che l'oggetto impiega per spostarsi da un punto ad un altro, cioè $t \geq \frac{1}{25} sec$, più sarà facile non solo vedere il movimento dell'oggetto ma anche ricostruire il movimento stesso a partire dai singoli frame del video.

A questo punto però bisogna capire qual è il movimento che vogliamo effettivamente essere in grado di vedere e ricostruire. Possiamo considerare due esempi per vedere quanto siano importanti le caratteristiche della telecamera utilizzata per registrare la scena.

- Se volessimo riuscire a vedere il tocco del piede a terra durante lo svolgimento dell'esame, dovremmo chiederci quanto velocemente avviene questo movimento.
- Se invece volessimo riuscire a vedere il tocco di un cono da parte dell'esaminando, dovremmo chiederci quanto potrebbe durare un impercettibile tocco del cono che poi ritorna nella sua posizione iniziale?

Nel caso in cui entrambi i movimenti citati sopra avvengano in un tempo $t \geq \frac{1}{25} sec$, cioè l'intero movimento avviene in un tempo maggiore di $0,04 sec$ allora saremo in grado di rilevarlo, perché rispetta quanto scritto fino ad ora.
Il problema di rilevamento del tocco del cono non si pone nel caso in cui dopo esser stato toccato cambia completamente posizione perché sempre in grado di capire cosa è effettivamente successo. Al contrario se dopo essere toccato tornasse nella sua posizione iniziale, il rilevamento di questo movimento richiederebbe capire quanto veloce il cono lo esegue e verificare di avere frame a sufficienza per renderci conto che quel movimento è avvenuto.
Per quanto riguarda il tocco del piede a terra, il suo rilevamento non è attualmente previsto in questo lavoro.

**Conclusioni intermedie.** Riassumendo quanto scritto fino ad ora, il nostro obiettivo era quello di ricostruire la traiettoria percorsa dalla moto lungo la pista. Si vuole vedere se la distanza percorsa dalla moto in $1/50\ sec$ sia abbastanza piccola da essere approssimata usando una linea retta per rappresentare lo spostamento della moto tra un frame ed il successivo, senza perdere informazioni.
Si consideri la velocità media calcolata nell'espressione 5.12, allora in $\Delta t = 1/50\ sec$ la moto riuscirà a percorrere una distanza pari a:

$$spostamento \approx v_m \cdot \Delta t \approx 8 \frac{m}{s} \cdot \frac{1}{50} sec \approx 0,16m = 16cm \tag{5.14}$$

Concludiamo affermando che partendo da un video registrato a 50fps, e supponendo che la moto si muova ad una velocità media di circa $8m/s$, abbiamo uno spostamento di circa $16cm$ tra un frame ed il successivo. Questa è una dimensione accettabile per approssimare il movimento della moto con segmenti lineari di 16cm.

Quindi il fatto di poter approssimare la traccia della moto con una serie di segmenti lineari ci permette di predirne il corso con il modello matematico citato nella sezione precedente, il filtro di Kalman (vedere sezione 3.5).

## 5.1 Omografia e calcolo matriciale

In questa fase del flusso di lavoro espresso nell'immagine 5.1 quello che si fa, come da titolo, è applicare l'omografia (vedere sezione 3.4 per approfondimento teorico). Come specificato nella sezione 3.4 uno dei tre casi in cui è possibile applicare il concetto di omografia riguarda: applicare una proiezione ad una superficie osservata da una telecamera in modo tale da ottenere la stessa superficie osservata però da un'altra telecamera o angolazione (vedere fig. 5.3).

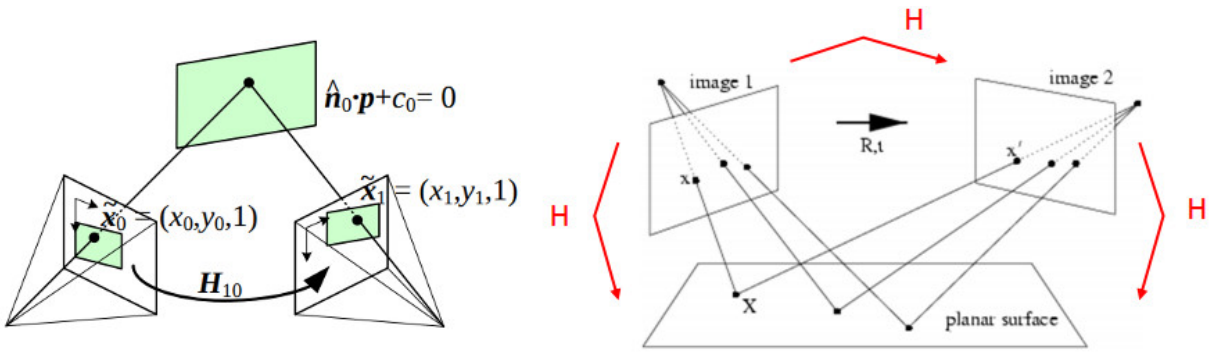



Figura 5.3

Figura 5.4: Visione della pista LSB dalla telecamera L2, pre applicazione dell'Omografia.

Quello dobbiamo fare nel nostro caso è proiettare la superficie vista dalla telecamera L2 (visibile in figura 5.4), che sarebbe lo spazio immagine in 2D, su un altra superficie in modo tale da

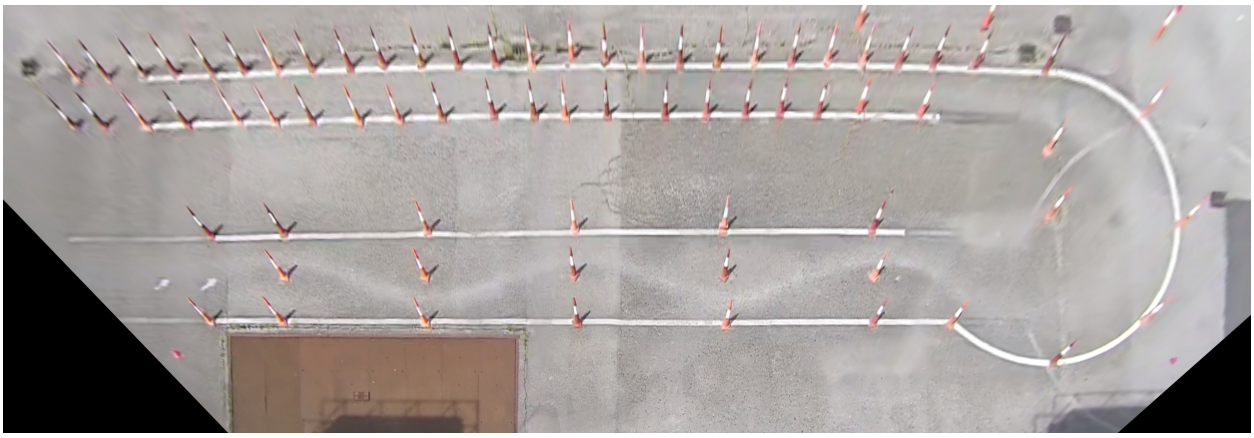


Figura 5.5: Visione della pista LSB dall'alto dopo aver applicato il processo di proiezione tramite omografia

ottenerne un'altra che simula la vista della pista LSB dall'alto (vedere figura 5.5). Si tenga conto del fatto che l'immagine su cui viene applicata l'omografia non è l'immagine originale ottenuta dalla telecamera bensì e frutto del processo di rettificazione del frame (vedere sezione 2.2).

**Calcolo matrice di Omografia.** Si analizza l'algoritmo usato per il calcolo della matrice di trasformazione (matrice di omografia). Se la matrice fosse già disponibile (calcolata) allora verrà recuperata dal file corrispondente senza la necessità di essere ricalcolata, a meno che la pista di riferimento cambi. Osserviamo come si comporta nel dettaglio l'algoritmo 2 "GeneraMatriceTrasformazione(FRAME)" usato per calcolare la matrice di omografia. Prima di tutto è necessario

**Algorithm 1** Gestione matrice di Omografia

```
1: if Il file Mat non esiste then
2:     matrix ← GeneraMatriceTrasformazione(FRAME)
3:     SalvaMatriceSuFile(matrix, PATHFILE)
4: else
5:     matrix ← LeggiMatriceDaFile(PATHFILE)
6: end if
```

recuperare il frame originario della telecamera L2 (nel nostro caso la figura 5.4). In seguito stabiliamo qual è la dimensione dell'immagine che vogliamo ottenere dopo l'applicazione dell'omografia, in questo caso sono state scelte delle dimensioni reali in centimetri, infatti "Size(3500,1200)" sta per $35$x$12$ metri che sono delle dimensioni leggermente più grandi rispetto alle dimensioni reali della pista LSB (come è possibile constatare nella piantina fornita dal ministero in figura 2.1, e la decreto [5]). A questo punto si definiscono prima di tutto dei punti sorgente nel frame in figura 5.4, ad esempio una serie di coni che uniti fra loro formino un poligono che racchiude l'intera pista. Poi è necessario definire dove questi punti sorgente verranno proiettati farlo ci si aiuta con le misure ufficiali direttamente fornite dal ministero, indicando le posizioni reali dei coni rispetto

**Algorithm 2** GeneraMatriceTrasformazione

```
1: function GeneraMatriceTrasformazione(mat: string, dimImmX: int, dimImmY: int) → Mat
2:     Leggi l'immagine sorgente della camera da mat;
3:     size_camera ← Size(dimImmX, dimImmY);
4:     im_dst_camera ← MatriceZeri(size_camera);
5:     pts_dst_camera ← [Punti destinazione];
6:     pts_src_camera ← [Punti sorgente], ▷ Selezionare i punti che nella realtà corrispondono ai punti specificati nell'immagine di destinazione
7:     h ← findHomography(pts_src_camera, pts_dst_camera);
8:     warpPerspective(im_src_camera, im_dst_camera, h, size_camera);
9:     return h
10: end function
```

ad un'origine prestabilita. Dopo aver scelto sia i punti sorgente che quelli di destinazione si utilizza un metodo direttamente fornito dalla libreria di Opencv "findHomography" che date delle corrispondenze tra punti trova la matrice di proiezione tra le due superfici (per riferimenti alla teoria di come si potrebbe ottenere la matrice di omografia vedere la sezione 3.4.1).

Al momento della scrittura di questa tesi il numero di punti sorgente scelti è fissato a 5, ma con opportune modifiche è possibile far sì che si possa scegliere una quantità arbitraria di coni di riferimento, con un minimo di 4 per avere una buona copertura della pista.

**Applicazione della matrice di Omografia.** Durante l'intero processo le immagini su cui si lavora sono considerate come delle matrici di interi da 8bit a 3 canali, cioè ogni cella della matrice rappresenta un colore. Dopo che la matrice di trasformazione è stata calcolata prima di tutto si applica al frame della telecamera, utilizzando anche qui un metodo direttamente fornito dalla libreria di OpenCV, così da ottenere il risultato che si può osservare in figura 5.5.

In seguito si passa ad operare sui percorsi della moto. Parliamo di percorsi al plurale perché prima di tutto è necessario fare una distinzione tra i percorsi dell'esaminato e quelli dell'istruttore, che useremo come riferimento per vari confronti che verranno fatti con il percorso dell'esaminato. Inoltre, ognuno dei due guidatori sarà caratterizzato a sua volta da due percorsi, perché la pista LSB, come abbiamo già scritto più volte, è ripresa da due diverse telecamere. La prima angolazione, riferita come L2 (visibile in figura 5.4), e la seconda, riferita come L1 (visibile in figura 5.6).

Figura 5.6: Visione della pista ripresa dalla telecamera L1.

**Algorithm 3** Trasformazione prospettica delle coordinate del percorso della moto

```
1: function perspectiveTransformation(h: Mat, percorsoMoto_prePT: vettore di punti2D,
   percorsoMoto_postPT: vettore di punti2D)
2:     for i ← 0 to dimensione di percorsoMoto_prePT do
3:         vettore di punti2D src
4:         vettore di punti2D dest
5:         Aggiungi percorsoMoto_prePT[i] a src
6:         perspectiveTransform(src, dest, h)
7:         Aggiungi dest[0] a percorsoMoto_postPT
8:     end for
9: end function
```

Il modo in cui otteniamo e sono rappresentati i percorsi della moto, sia dell'esaminatore che dell'esaminato, è specificato nella sezione 2.3. Si analizza adesso l'algoritmo che si occupa della trasformazione del percorso della moto applicando la matrice di Omografia calcolata in precedenza. L'elemento principale ed essenziale dell'algoritmo 3 è la funzione "perspectiveTransform(src, dest, h)" (anche questo metodo è fornito dalla libreria di OpenCV). Quello che fa questa funzione è moltiplicare il vettore che rappresenta una posizione 2D nell'immagine di partenza con la matrice di omografia, così da applicare la trasformazione ai punti passati come argomento, e restituire così un nuovo insieme di punti, dove ognuno di essi è un punto nello spazio generato con la proiezione applicata grazie alla matrice "h". Come si vede nell'algoritmo, è stato usato un ciclo "For" per processare tutti i punti del percorso, in realtà il metodo stesso è in grado di processare sia in input che in output degli array, quindi non è realmente necessario passare un punto per volta. Al termine della funzione nel vettore di punti "PercorsoMoto_postPt" avremo il percorso della moto con la trasformazione applicata, ed è possibile apprezzarne il risultato nella figura 5.8. Si osservi come l'identità del percorso è rimasta la stessa, la differenza sta nel fatto che adesso è stato proiettato su uno spazio diverso.


Figura 5.7: In figura si può vedere il percorso originale della moto originale rappresentato in coordinate immagine 2D non ancora trasformate.

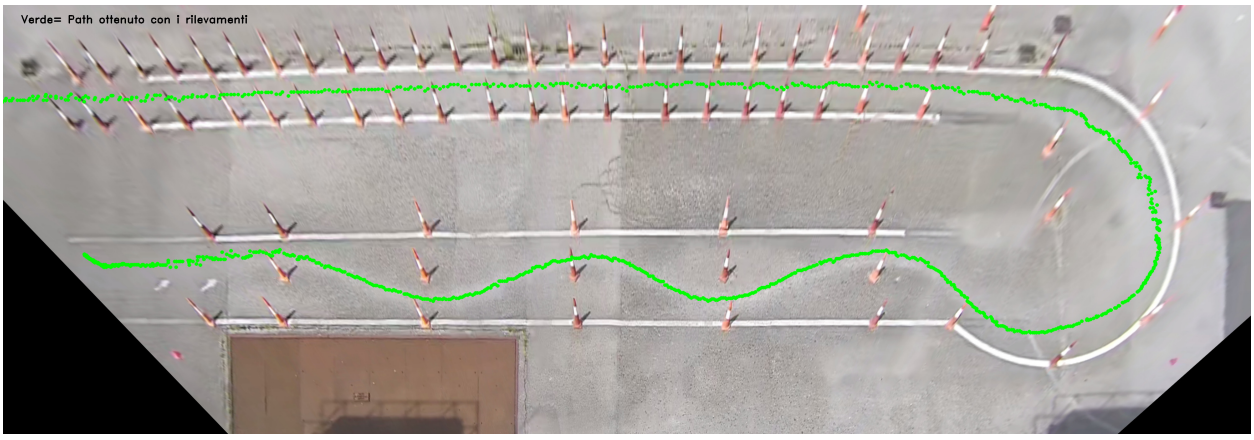



Figura 5.8: In figura è rappresentato il percorso della moto dopo che la trasformazione è stata applicata.

## 5.2 Utilizzo del Filtro di Kalman

Quando si cerca di recuperare le coordinate dei vari oggetti sparsi per il frame attraverso le informazioni date sui Bounding Box (per spiegazione più dettagliata si faccia riferimento alla sezione 2.3) bisogna tenere conto di due situazioni particolari:

1. il Bounding Box che circonda la moto, oppure un cono, potrebbe essere completamente assente, anche se la moto sta correttamente continuando il suo percorso oppure il cono non si è minimamente mosso (**Falso negativo**);

2. il Bounding Box potrebbe trovarsi un una posizione completamente diversa dalla posizione reale della moto, oppure del cono (**Falso Positivo**).

Riferendoci esclusivamente al tracciamento della moto, lo strumento che utilizziamo per ovviare alla presenza dei due possibili problemi citati sopra è il Filtro di Kalman (alla sezione 3.5 è possibile osservare nel dettaglio i vari elementi che lo compongono). Lo scopo del suo utilizzo sta nel prevedere quale cammino potrebbe fare la moto mitigando eventuali di errori dovuti a falsi positivi o negativi dovuti al rilevamento della moto, e tenendo conto del percorso fatto e di cosa potrebbe accadere nell'immediato futuro.

**Algorithm 4** Applicazione del filtro di Kalman sui percorsi 3D post proiezione prospettica

**function** KALMANFILTERAPPLICATION(percorsoMoto3D_postProiezione: vettore di punti3D, percorsoMoto3D_postProiezione: vettore di punti3D)
    Crea un filtro di Kalman $KF$ con dimensioni $6 \times 3$
    $\Delta t \leftarrow 0.02$ ▷ Intervallo di tempo fra una misurazione e la successiva
    Imposta matrice di transizione $\mathbf{F_k}$ come matrice identità e inserendo $\Delta t$ nei posti giusti
    Imposta $\mathbf{B_k}$ (matrice di controllo) inserendo $\frac{(\Delta t)^2}{2}$ e $\Delta t$ nei posti giusti.
    Imposta la matrice di covarianza di rumore del processo $\mathbf{Q_k}$ come matrice identità con elementi $1 \times 10^{-2}$
    Imposta la matrice di osservazione $\mathbf{H_k}$ come matrice identità
    Imposta la matrice di covarianza dell'osservazione $\mathbf{R_k}$ come matrice identità con elementi $0.1$
    Crea una matrice $measurement$ di dimensione $3 \times 1$ inizializzata a $0$
    **for** $i \leftarrow 0$ to dimensione di $percorsoMoto3D\_postProiezione$ **do**
        $measurement \leftarrow percorsoMoto3D\_postProiezione[i]$
        **if** $i = 0$ **then**
            $x_k \leftarrow measurement$
        **end if**
        $predizioneMisura \leftarrow KF.predict()$
        $MisuraAggiornata \leftarrow KF.correct(measurement)$
        Aggiungi $MisuraAggiornata$ a $PercorsoMoto3D\_postKalmaFilter$
    **end for**
**end function**

L'effettiva implementazione del Filtro di Kalman, con tutte le moltiplicazioni tra matrici e vettori che è necessario fare durante i passi di predizione e aggiornamento (espressamente esplicate nella sezione 3.5) è fornita direttamente dalla libreria di OpenCV. Bisogna solamente inizializzare gli elementi che lo compongono in base al risultato che si vuole ottenere.

1. Prima di tutto è necessario definire da quanti elementi è composto lo stato $\mathbf{x_k}$. Nel nostro caso usiamo un vettore di sei elementi, cioè tre elementi per indicare una posizione 3D, mentre gli altri tre per indicare le componenti della velocità della moto lungo i 3 assi. Sia per la posizione che per la velocità la componente $z$ sarà sempre 0, visto che tutto il percorso avviene ad altezza 0. Le misurazioni sono a loro volta un vettore di tre elementi che rappresenta la posizione della moto nello spazio di riferimento.

2. Poi si passa all'inizializzazione della matrice di transizione $\mathbf{F_K}$ e alla matrice di controllo $\mathbf{B_K}$ come esplicato nella sezione 3.5.

3. In seguito si inizializzano come matrici identità la matrice di osservazione $\mathbf{H_k}$, mentre la matrice di covarianza dell'osservazione $\mathbf{R_k}$ con $0.1$ come elementi sulla diagonale principale ed il resto 0.

4. L'ultimo elemento da inizializzare è la matrice di covarianza del rumore di processo $\mathbf{Q_k}$, e viene inizializzata con il valore $1x10^{-2}$ sulla diagonale.

La dimensione di tutte la matrici che compongo il filtro dipendono dalla dimensione dello stato $\mathbf{x_k}$. Tutte le dimensioni devono combaciare per evitare che ci siano problemi nelle moltiplicazioni e somme tra vettori nelle formule necessarie alle fasi di predizione e aggiornamento (per vedere la varie formule nel dettaglio vedere sezione 3.5). Ad esempio se il nostro stato fosse composto da sei elementi, allora matrice di transizione dovra essere una matrice $6x6$, quella di osservazione sarà una $3x6$, mentre quella di controllo sarà una $6x3$. Anche tutti i vettori impegnati nei vari calcoli avranno delle dimensioni compatibili alle matrici.

E' necessario dire qualcosa di più riguardo alla matrice $\mathbf{Q_k}$ e ciò che essa rappresenta, cioè quanto sono rumorose le misurazioni fornite al filtro di Kalman come argomento. Per meglio dire, più basso è il valore con cui inizializziamo la matrice $\mathbf{Q_k}$, maggiore sarà il lavoro di riduzione del rumore sulle predizioni e aggiornamento della predizione che il filtro andrà a fare. Invece più è alto il valore di inizializzazione più saranno simili fra loro le misurazioni originali e il risultato dell'applicazione del filtro, ciò sta a significare che i dati iniziali sono poco rumorosi. Purtroppo per mancanza di tempo durante lo sviluppo del sistema non c'è stata la possibilità di fare dei calcoli appropriati per determinare con precisione quale è il rumore che affligge le nostre misurazioni, quindi dopo vari tentativi con diversi valori, suggeriti dalla stessa documentazione fornita da OpenCV, si è optato per lasciare il valore di default suggerito di $0.01$. Grazie a questo valore riusciamo a raggiungere l'obiettivo mostrato all'inizio della sezione corrente apprezzabile in figura 5.9.

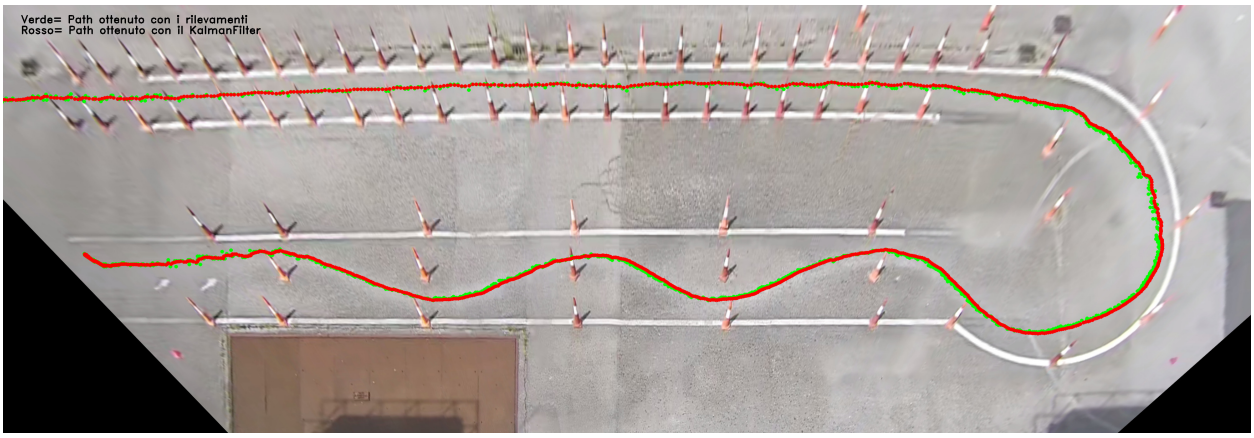



Figura 5.9: Si noti il risultato dell'applicazione del filtro di Kalman. Il percorso in verde è il percorso rilevato originariamente affetto da rumore e possibili errori di rilevamento, in rosso invece abbiamo il percorso ottenuto con il filtro di Kalman.

Analizzando attentamente la figura 5.9 si veda come il percorso rosso (risultato del filtro di Kalman) sia molto più armonioso rispetto a quello verde (rilevazione originaria del percorso dopo aver applicato la proiezione). Modificando i valori di inizializzazione dei vari elementi che compongo il filtro è possibile ottenere risultati molto diversi fra loro. Esiste anche la possibilità di far considerare al filtro non solo la velocità della moto, ma anche la sua accelerazione, quantità entrambe calcolabili avendo il percorso rilevato della moto.

## 5.3 Calcolo della spline per l'armonizzazione dei dati

Analizzando attentamente l'immagine 5.9, nonostante l'applicazione del Filtro di Kalman abbia fatto un buon lavoro nel raggiungere il suo obiettivo principale (specificato nella sezione precedente), aveva in parte anche un secondo fine che era quello di diminuire il rumore che affliggeva le rilevazioni del percorso della moto. Guardando l'immagine 5.9 possiamo dire che in parte c'è anche riuscito, ma è ritenuto sufficiente. Il motivo è che il percorso della moto verrà utilizzato per una rappresentazione virtuale dell'esame in Unreal Engine (che verrà esplicata nel dettaglio successivamente). Riguardo a questa rappresentazione virtuale possiamo dire con certezza che minore è il rumore che affligge i dati, migliore sarà la rappresentazione virtuale. Quindi per ottenere un percorso ancora più armonioso e il più possibile privo di rumore usiamo le spline.

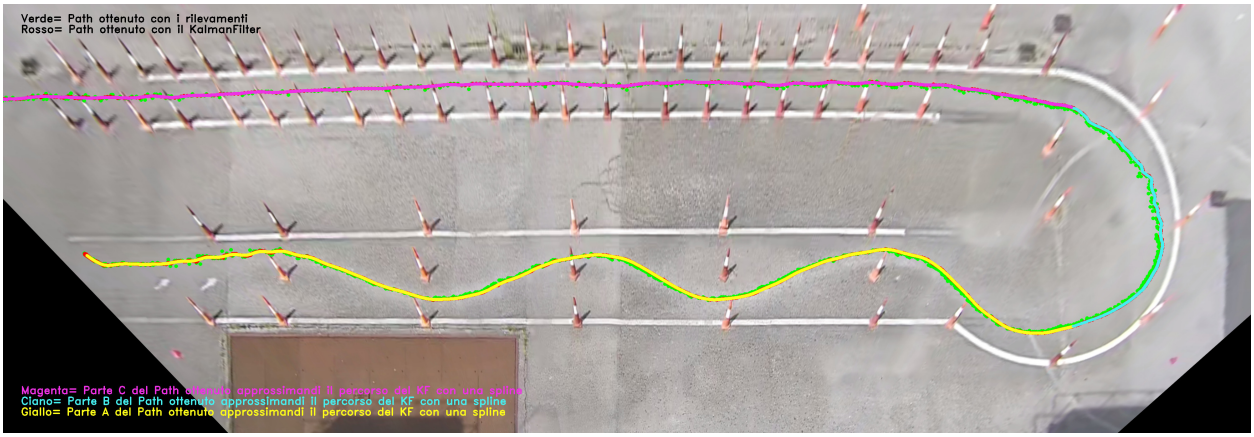



Figura 5.10: Percorso risultante dopo il calcolo della spline. Come si può ben vedere in realtà le spline sono tre, ognuna per un pezzo del percorso.

Il processo mentale che ci ha portato alla scelta di questa tecnologia è il seguente. Come descritto nelle sezione precedenti, i dati di partenza ottenuti dalle rilevazione della moto in movimento sulla pista LSB sono rumorosi, (non abbiamo una misura precisa di questa rumorosità a causa di mancanza di tempo), e noi vogliamo in un certo senso levigare, o se possibile rimuovere questa rumorosità. Come si legge alla sezione 3.6, le spline sono "funzioni utilizzate in applicazione che richiedono l'interpolazione e/o il livellamento (approssimazione) dei dati". Due proprietà di una spline sono le seguenti:

- **Interpolazione**: proprietà per cui una spline passa attraverso un insieme specificato di punti di dati. Le spline interpolanti sono utilizzate per rappresentare dati in modo preciso. Garantiscono che la curva passi esattamente attraverso i punti noti, fornendo una stima accurata dei valori intermedi tra i vari punti forniti.
- **Approssimazione dei dati**: proprietà simile all'interpolazione, ma meno vincolante. Le spline approssimanti cercano di modellare una curva che si avvicini ai dati senza necessariamente passare attraverso tutti i punti. Questa proprietà è preziosa quando i dati possono

contenere rumore o incertezze, poiché le spline possono estrarre il modello sottostante senza essere influenzate da singoli punti anomali.

Leggendo quanto appena scritto, ed in particolare la proprietà di **Approssimazione dei dati**, ci rendiamo conto che è proprio quelli di cui abbiamo bisogno, una approssimazione del percorso della moto post applicazione del filtro di Kalman che non passi necessariamente per i punti dati e che cerchi di ridurre il più possibile rumori e incertezze; e queste sono le motivazioni che ci hanno portato alla scelta di utilizzare le spline.

Si analizza adesso l'algoritmo 5 che si occupa del calcolo della spline. Per generare la spline sono stati usati i metodi forniti dalla libreria ALGLIB ([39]), in particolare le funzioni **spline1dfit** e **spline1dcalc**.

**spline1dfit(X,Y,N,M,LambdaNS).** Questa funzione approssima $N$ punti sparsi (alcuni punti di X[] possono anche essere uguali tra loro) tramite una spline cubica con M nodi equidistanti in un intervallo $[min(x), max(x)]$. L'output della funzione è un oggetto $S$ che rappresenta la spline. Vediamo invece alcuni input della funzione:

- **X**= punti, array[0...N-1]
- **Y**= valori di funzione, array[0...N-1]
- **N**= numero di punti (opzionale):
  - $N > 0$
  - Se dato, solo i primi $N$ elementi di X e Y sono processati
  - Se non dato, viene automaticamente determinato dalla lunghezza
- **M**= numero di funzioni di base (=numero dei nodi), $M \geq 4$
- **LambdaNS**= $LambdaNs \geq 0$, è una costante di regolarizzazione. Penalizza la non linearità nella spline di regressione.

**spline1dcalc(S,X).** Quello che fa questa funzione è semplicemente calcolare il valore della spline al dato punto X. Non è altro che un'applicazione della funzione spline al valore passato come argomento.

**Algorithm 5** Calcolo della spline

**function** GENERAPERCORSOSPLINE(percorsoMoto3D_postKalmaFilter: vettore di punti3D) $->$ vettore di punti3D che rappresenta la spline
    Divido il percorso in percorsoMoto3D_postKalmaFilter in tre sezioni differenti A, B e C
    splineA $\leftarrow$ **spline1dfit**(xA, yA, A.length, nodi_spline1, $0.00001$)
    splineB $\leftarrow$ **spline1dfit**(yB, xB, B.length, nodi_spline2, $0.00001$)
    splineC $\leftarrow$ **spline1dfit**(xC, yC, C.length, nodi_spline3, $0.00001$)
    **for** i $\leftarrow$ A.start to A.fine **do**
        x $\leftarrow$ A[i]
        y $\leftarrow$ **spline1dcalc**(splineA, x)
        Aggiunge il punto $(x, y, 0)$ alla fine del vettore $percorsoMoto3D\_postSpline$
    **end for**
    **for** i $\leftarrow$ B.start to B.fine **do**
        y $\leftarrow$ B[i]
        x $\leftarrow$ **spline1dcalc**(splineB, y)
        Aggiunge il punto $(x, y, 0)$ alla fine del vettore $percorsoMoto3D\_postSpline$
    **end for**
    **for** i $\leftarrow$ C.start to C.fine **do**
        x $\leftarrow$ C[i]
        y $\leftarrow$ **spline1dcalc**(splineC, x)
        Aggiunge il punto $(x, y, 0)$ alla fine del vettore $percorsoMoto3D\_postSpline$
    **end forreturn** $percorsoMoto3D\_postSpline$
**end function**

Si cerca adesso di dare una giustificazione e spiegazione a quanto fatto nell'algoritmo appena scritto. Osservando sia l'immagine 5.10 che l'algoritmo 5 è possibile notare come il percorso che la generazione della spline siano state divise in 3 sezioni A, B e C. Il motivo di questa azione è dovuto all'utilizzo di una spline monodimensionale. Sarebbe naturale pensare che con una spline bidimensionale non sarebbe stato necessario attuare questa divisone in sezioni; però nonostante la stessa libreria ALGLIB fornisse un metodo per il calcolo di una spline bidimensionale a causa di mancanza di tempo e della possibilità di testare i risultati ottenibili con una spline bidimensionale, si è optato per l'utilizzo di una spline monodimensionale (il cui metodo era già conosciuto e testato), e questa scelta ha portato alla divisione necessaria della spline in 3 sezione per evitare che i punti appartenenti a sezioni diverse venissero approssimati tra loro durante il calcolo delle spline nullificando tutto il lavoro fatto.

Teniamo conto che guardando l'immagine 5.10 l'asse x è posto orizzontalmente, e cresce in valore verso destra, mentre l'asse y è posto verticalmente all'immagine e cresce verso il basso. Un altra cosa che potrebbe saltare all'occhio analizzando l'algoritmo 5, riguarda il fatto che nella sezione B del percorso come variabile di riferimento della spline non è stata usata la x bensì la y. Il motivo di tale scelta è dovuto al fatto che nella sezione B del percorso l'intervallo di valori in cui varia y rispetto all'intervallo di valori in cui varia x è di molto maggiore, e per come si

comporta il metodo **spline1dfit** dopo vari test si è ritenuto fosse meglio usare la coordinata y come riferimento. Essendo di più i valori di y su cui calcolare la spline, il metodo aveva la possibilità di generare molti più nodi attraverso cui far passare la funzione, cosa che invece non sarebbe accaduta in egual misura nel caso in cui si sarebbe fatto al contrario. Nelle sezioni A e C invece accade tutto il contrario, e per questo motivo in quelle zone si usa x piuttosto che y.

Per quanto riguarda invece il metodo di divisone del percorso nelle tre sezioni A, B e C anche qui è possibile dire qualcosa. Il metodo scelto per la divisione in sezioni non è assolutamente il migliore, infatti il metodo usato è stato semplicemente quello di osservare diversi percorsi per vedere a che punto iniziasse la curva e tagliare il percorso, per la fine invece controllavamo dove finisse la curva. Questo è un metodo molto grezzo e assolutamente inefficiente per portare a termine il lavoro pensato a causa di mancanza di tempo, infatti si era pensato ad una possibile soluzione che di sicuro ci permetterebbe di ottenere risultati migliori rispetto al metodo attualmente in uso. Essa consisteva nel controllare le derivate punto per punto e se per un certo intervallo il suo valore continuava a crescere allora voleva dire che ci trovavamo all'inizio della curva, mentre se succedeva il contrario ci trovavamo alla fine della curva; utilizzando una soluzione del genere bisognava stare attenti a non tagliare il percorso nel bel mezzo dello slalom, ma sicuramente la divisione in sezioni del percorso sarebbe risultata più naturale e dipendente dal tipo di percorso. Però per mancanza di tempo e quindi possibilità di testare la soluzione alla fine si è optato per mantenere la soluzione in uso.

In figura 5.11 si osservare l'effettivo risultato ottenuto dopo aver unito le 3 spline che compongono il percorso finale.

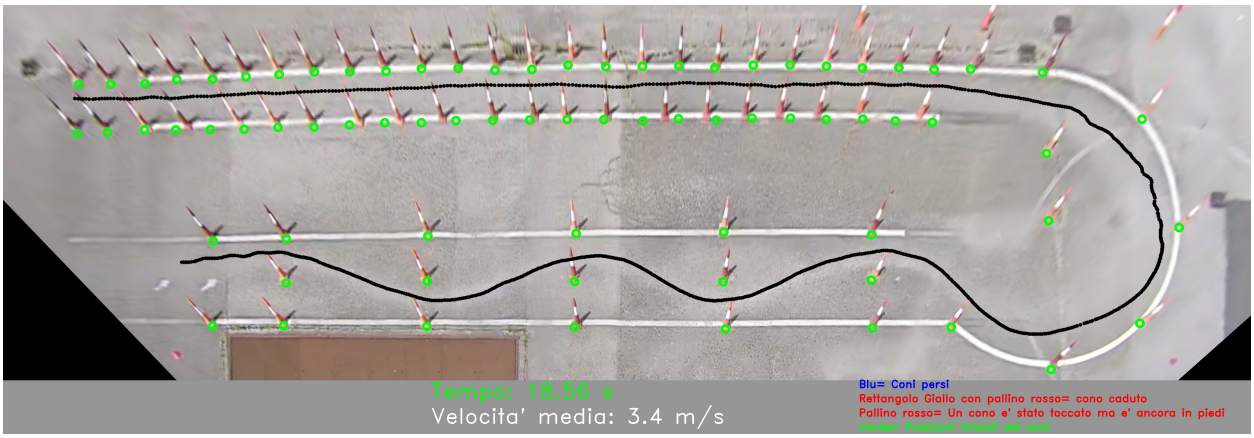



Figura 5.11: Risultato del ricampionamento della Spline

## 5.4 Calcolo del punteggio del percorso

L'obiettivo di un esame di patente di guida è sicuramente quello di verificare che l'esaminato possegga le capacità necessarie per mettersi alla guida di un veicolo senza rappresentare un pericolo per gli altri. Come già specificato nella sezione 2 il nostro contesto di analisi è l'esame di patente della moto, ed in particolare l'analisi del percorso sulla pista LSB. Secondo la legge, per passare questa fase dell'esame è necessario eseguire il percorso LSB in un tempo non minore di 15 secondi e ovviamente non bisogna commettere altri errori durante il percorso. Secondo quanto ci è stato detto dall'esaminatore certificato, che ha svolto quasi la totalità delle guide usate come riferimento per i vari risultati mostrati come immagini e dati durante questo lavoro di relazione, non basta completare il percorso correttamente in un tempo maggiore di 15 secondi ma è anche necessario che l'esaminatore reputi l'esaminato adatto alla guida, cioè l'esaminato non deve mostrare:

- guida coordinata irregolarmente dimostrando scarsa abilità;
- scarsa abilità nella gestione della velocità del veicolo;
- postura scorretta durante la guida.

Per questo motivo abbiamo provato a codificare alcune di queste richieste utilizzando i dati che abbiamo a disposizione, e fornendo all'esaminatore un sistema di confronto del percorso che va al di là del semplice risultato promosso/bocciato, fornendo indicazioni utili e misurabili di tipici errori d'esame e procedure di guida non sicure.

Il sistema di confronto ideato è molto semplice, e si basa principalmente sul confrontare i due percorsi:

1. Il primo sarà un percorso fatto dall'esaminatore, e che verrà identificato come il percorso ottimo.
2. Il secondo sarà invece un normale percorso fatto dall'esaminato.

Il punteggio verrà calcolato per entrambi i percorsi, così poi da poterne confrontare le caratteristiche. Non è necessario che il percorso fatto dell'esaminatore venga analizzato e processato ogni singola volta, si può benissimo scegliere un percorso che viene considerato ottimo, e prenderlo come riferimento senza più cambiarlo.

Il punteggio finale del percorso è un valore compreso tra 0 e 1, dato dalla moltiplicazione dei punteggi calcolati sulle diverse caratteristiche del percorso e della guida dell'esaminato che cerchiamo di tenere sotto analisi; elenchiamo adesso quali sono queste caratteristiche:

1. area esterna alla traccia definita dal percorso, spigheremo più avanti cosa si intende;
2. traiettoria dell'esaminato e somiglianza con quella dell'esaminatore;
3. tempo di esecuzione del percorso;
4. controllo della velocità del motociclo da parte dell'esaminato;

La formula utilizzata per generare un punteggio relativo ad una delle caratteristiche specificate nell'elenco precedente è la seguente:

$$punteggio = \frac{|val\_istruttore|}{|val\_istruttore| + (|val\_istruttore - val\_esaminato|)} \tag{5.15}$$

Il punteggio finale, dato un insieme di $punteggio$, $\{punteggio_i, ..., punteggio_n\}$, supponendo che siano $n$ le caratteristiche codificate, sarà dato dalla seguente formula:

$$punteggio\_finale = \frac{1}{n} \cdot \sum_{i=1}^{n} punteggio_i \cdot \alpha_i; \tag{5.16}$$

dove ogni $\alpha_i$ è un peso assegnato allo specifico punteggio di una caratteristica del percorso. L'idea a cui abbiamo pensato per la definizione di questi pesi era quella di avere un set di dati (diversi percorsi con diverse caratteristiche) abbastanza grande da poter allenare una rete neurale implementata ad hoc, il cui allenamento sarebbe stato supervisionato dall'istruttore stesso, definendo la bontà di un percorso. L'output di questo processo di allenamento sarebbero stati i pesi da usare nella formula 5.16 di calcolo del punteggio finale. Questa idea però richiede una quantità non indifferente di percorsi e dati che non è stato possibile raccogliere nei tempi del tirocinio; quindi alla fine abbiamo deciso lo stesso di calcolare i punteggi separati sulle varie caratteristiche del percorso, ma non dare un punteggio finale che non sarebbe stato rappresentativo delle caratteristiche dello stesso.
Considerando la formula 5.15 vale sempre che $0 \leq punteggio \leq 1$. Questo perché più il valore di $val\_esaminato$ è distante da $val\_istruttore$ in modulo, maggiore sarà il valore del denominatore e quindi il valore di un punteggio sarà sempre più vicino allo 0. Al contrario, più $val\_esaminato$ è vicino in modulo a $val\_istruttore$ più basso sarà il valore del denominatore (anche se sarà sempre maggiore del numeratore) e quindi il punteggio sarà sempre più vicino a 1. Si deduce che più il valore del $punteggio$ è vicino a 1, migliore è il punteggio ottenuto in una specifica categoria. Adesso si analizzano nel dettaglio i vari elementi che cerchiamo di tenere sotto osservazione, e quali punteggi generano.

**Area attorno alla traiettoria e sua analisi.** Specifichiamo prima di tutto cosa intendiamo con area esterna alla traccia. Come si può notare in figura 5.12 è presente una zona bianca, e questa zona e rappresenta l'area di cui si parlava.

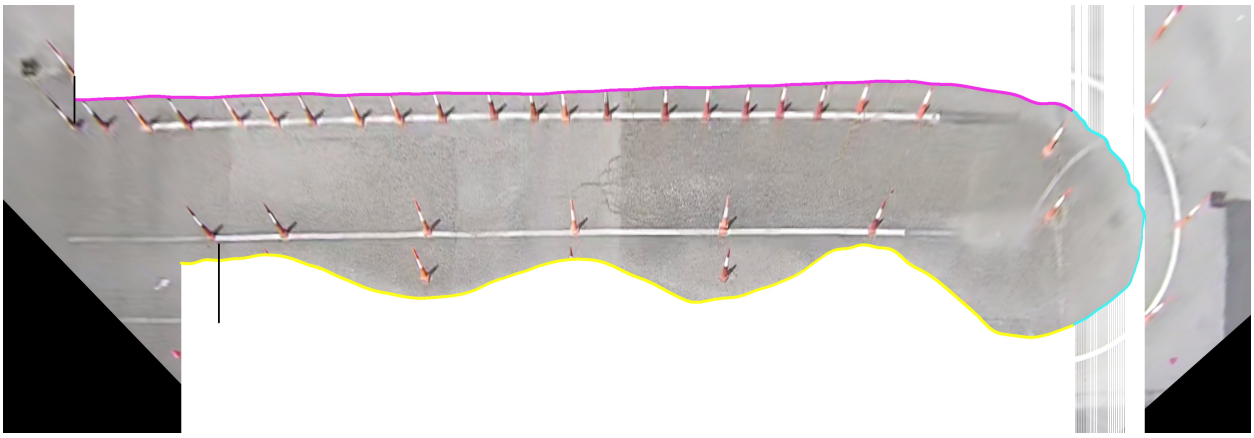


Figura 5.12: In bianco è possibile vedere l'area che genera il percorso attorno a se.

Abbiamo pensato fosse un buon modo per analizzare la traiettoria della moto per vari motivi:

- ad esempio, una traiettoria che nel rettilineo finale si spinge molto verso l'esterno rischia di prendere i birilli e cadere. In questo caso l'area attorno alla traiettoria eseguita dall'esaminato risulterà diversa da quella dell'esaminatore che idealmente passa al centro perfetto del rettilineo. Vale lo stesso nel caso in cui l'esaminato passa troppo vicino all'altro lato del rettilineo.
- Un altro esempio potrebbe essere una curva eseguita troppo stretta, con il rischio di cadere, oppure una curva eseguita troppo larga con il rischio di toccare i coni esterni o uscire dalla pista. Anche qui in base al tipo di curva eseguita l'area potrebbe cambiare molto.
- Anche un slalom troppo stretto o troppo largo potrebbero influire rispettivamente sul toccare un cono oppure sull'uscire di pista.

Utilizziamo quindi il valore di quest'area per determinare se abbiamo una buona traiettoria oppure no. Il metodo utilizzato per calcolarla è molto semplice. Sappiamo già le coordinate della traiettoria della moto, e sappiamo anche che la dimensione dell'immagine su cui andiamo a proiettare i percorsi è sempre la stessa (vedere sezione 5.1), allora per calcolare l'area (la zona bianca in figura 5.12) andremo semplicemente a fare una sommatoria delle distanze tra la coordinata y di un punto e la dimensione massima dell'immagine. Dopo aver ottenuto sia l'area generata dalla traiettoria dell'istruttore che dell'esaminato, si utilizza semplicemente la formula 5.15 per calcolare il punteggio relativo all'area, mettendo le aree calcolate al posto giusto.

Siamo ben consapevoli che questa quantità da sola non sia in grado di caratterizzare bene una buona guida da parte dell'esaminato, per questo motivo viene affiancata ad un altro tipo di punteggio. L'idea si basa sull'usare il coefficiente di correlazione, e adesso cercheremo di spiegarne il perché. Sappiamo che la traiettoria della moto è formata da una serie di punti 2D (x,y), allora quello che andremo a fare e calcolare il coefficiente di correlazione sulle coordinate x dei percorsi

dell'istruttore e dell'esaminato e sulle coordinate y, separatamente.

$$coeff\_corr(x_{istruttore}, x_{esaminato}) = \frac{\mathbf{cov}(x_{istruttore}, x_{esaminato})}{\sigma(x_{istruttore}) \cdot \sigma(x_{esaminato})} \tag{5.17}$$

$$coeff\_corr(y_{istruttore}, y_{esaminato}) = \frac{\mathbf{cov}(y_{istruttore}, y_{esaminato})}{\sigma(y_{istruttore}) \cdot \sigma(y_{esaminato})} \tag{5.18}$$

Come ben sappiamo (vedere sezione 3.1.1), il coefficiente di correlazione è un valore compreso sempre tra 0 e 1 in modulo, quindi è una valore adatto per far parte del punteggio finale. Partiamo dicendo che più i valori di entrambi i coefficienti sono vicini a 1, più le traiettorie dell'istruttore e dell'esaminato saranno simili, e cerchiamo di capirne il perché.
La formule della covarianza e della deviazione standard o scarto quadratico medio ($\sigma$) sono le seguenti (valori barrati rappresentano la media campionaria):

$$\mathbf{cov}(x_{istr}, x_{esam}) = \frac{1}{(n-1)} \sum_{i=1}^{n} (x\_istr_i - \overline{x\_istr}) \cdot (x\_esam_i - \overline{x\_esam}) \tag{5.19}$$

$$\sigma(x\_istr) = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^{n} (x\_istr_i - \overline{x\_istr})^2} \tag{5.20}$$

$$\mathbf{cov}(y_{istr}, y_{esam}) = \frac{1}{(n-1)} \sum_{i=1}^{n} (y\_istr_i - \overline{y\_istr}) \cdot (y\_esam_i - \overline{y\_esam}) \tag{5.21}$$

$$\sigma(y\_istr) = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^{n} (y\_istr_i - \overline{y\_istr})^2} \tag{5.22}$$

Analizzando entrambe le formule appena scritte e sapendo che la $cov(x, x) = var(x)$ con x campione di dati $x = \{x_1, ..., x_n\}$ e che $Var(x) = \sigma(x)^2$, possiamo notare questo. Maggiore e la relazione lineare che c'è tra le coordinate x dell'istruttore e dell'esaminato più vicino a 1 sarà il valore del coefficiente di correlazione. Se la medesima cosa avviene anche per le coordinate y allora possiamo assumere che i due percorsi abbiano una traiettoria molto simile, anzi più i due coefficienti di correlazione sono vicini a 1 più le due traiettorie sono sovrapponibili.

**Tempo esecuzione percorso.** Come già scritto più volte la nostra pista di riferimento è la LSB, è per passare questa fase dell'esame è necessario che l'esaminato la percorra senza errori in un tempo minimo di 15 secondi. Sempre da quanto ci è stato detto da un istruttore è vero che il tempo minimo è di 15 secondi ma questo non vuol dire che il percorso possa essere completato in un tempo esageratamente alto. Per questo motivo abbiamo pensato di aggiungere un punteggio che riguardasse il tempo di esecuzione del percorso, infatti più il tempo dell'esaminato si avvicina a quello dell'istruttore maggiore sarà il punteggio ottenuto, ma si parla sempre di un valore compreso tra 0 e 1.

Per quanto riguarda il calcolare il tempo che la moto impiega per effettuare il percorso, il procedimento è molto più facile di quello che sembra. Sappiamo già che la nostra telecamera registra

a 50 fps, è che quindi è come se scattasse una foto della realtà ogni 0,02 secondi. Questo vuol dire che tra una punto 2D della nostra traiettoria ed il successivo passano proprio 0,02 secondi, allora quello che ci rimane da fare è semplicemente moltiplicare il numero di punto della traiettoria per 0,02 ed otteniamo il tempo che ha impiegato l'esaminato per completare il percorso.

Un piccolo appunto va fatto sui punti della traiettoria che consideriamo per calcolare il tempo di completamento del percorso. In realtà non consideriamo tutte le posizioni che abbiamo a disposizione, bensì iniziamo a contare il tempo nel momento in cui, idealmente, la ruota anteriore della moto oltrepassa i primi coni della pista e ci fermiamo quando avviene lo stesso alla fine del tracciato.

Dopo che il tempo è stato calcolato, si utilizza semplicemente la formula 5.15 per determinare il punteggio ottenuto confrontando i tempi dell'istruttore e dell'esaminato.

**Controllo della velocità.** Un altra serie di punteggi vengono calcolati in base al controllo che l'esaminato ha della velocità della moto, e definiamo secondo quale criterio generiamo questi punteggi e perché. Prima di tutto parliamo della velocità come grandezza, ed in seguito vederemo anche come poter utilizzare l'accelerazione per ottenere alcune informazioni sulla guida. Grazie ai dati in nostro possesso siamo in grado di calcolare sia la velocità che l'accelerazione, visto che siamo in possesso delle varie posizioni lungo la traiettoria e l'intervallo di tempo $\Delta t$ che c'è tra una posizione e la successiva.

Avere un buon controllo della velocità durante il percorso è sicuramente sinonimo di aver fatto una buona guida, per questo motivo cerchiamo di analizzale al guida dell'esaminato usando queste tre grandezze:

- **velocità minima**. Una velocità minima troppo bassa potrebbe indicare che la moto si è fermata anche se per un attimo, e sappiamo che da regolamento fermarsi durante il percorso è una penalità. Invece, una velocità minima troppo alta potrebbe indicare che l'esaminato è andato troppo veloce durante il percorso.

- **Velocità massima**. Così come la velocità minima, una velocità massima troppo alta potrebbe indicare che l'esaminato ha eseguito il percorso troppo velocemente.

- **Velocità media**. Anche questa quantità è in grado di dirci qualcosa, ad esempio, nel caso in cui fosse troppo grande allora vuol dire che l'esaminato ha mantenuto una velocità elevata durante il percorso e probabilmente potrebbe aver terminato il percorso in meno di 15 secondi.

Queste tre quantità prese singolarmente non ci dicono poi molto sulla guida dell'esaminato però combinate insieme ci permettono di trarre diverse in formazioni, dopo averle confrontate con le velocità ottenute dal percorso dell'istruttore (usando sempre la formula 5.15).

Un lavoro simile lo facciamo con le accelerazioni. Anche queste grandezze sono in grado di dirci qualcosa sulla guida dell'esaminato.

- **Accelerazione minima**. Sappiamo che quando l'accelerazione è negativa allora c'è stata una diminuzione di velocità, però se questo valore negativo fosse troppo basso allora vorrebbe dire che, si c'è stata una diminuzione di velocità però al tempo stesso è stata molto veloce, e nella realtà questo consiste praticamente in una frenata brusca, che sicuramente non è rappresentativa di una buona guida.
- **Accelerazione massima**. Si possono fare delle considerazioni simili anche per questa grandezza. Un accelerazione massima troppo grande mentre magari si ha un'accelerazione quasi costante durante tutto il percorso vorrebbe dire che l'esaminato ha accelerato improvvisamente in qualche punto.
- **Accelerazione media effettiva**. Di base visto che le accelerazioni possono anche essere negative, se andassimo a calcolare la media delle accelerazioni lungo un percorso con un inizio ed una fine in teoria questa dovrebbe venire 0, visto che la moto parte da ferma, e alla fine del percorso ritorna ad essere ferma. Per evitare questo non calcoliamo la media normalmente, bensì eleviamo al quadrato ogni accelerazioni considerata, in questo modo il risultato finale non sarà 0. Avere un'accelerazione media effettiva che si distanzia troppo da quella dell'istruttore non è un sintomo di una buona guida.

Anche in questo, i tre punteggi relativi alle accelerazioni presi singolarmente non ci dicono poi molto sulla guida dell'esaminato, però tutte insieme, combinate anche ai punteggi sulla velocità potrebbero permetterci di ottenere delle informazioni affidabili sulla guida sostenuta.

## 5.5 Analisi correttezza traiettoria

Fino ad ora ci siamo preoccupati di analizzare la forma della traccia definita dal percorso eseguito dall'esaminato, e alcune sue caratteristiche ad esempio: la forma, vari tipi di velocità, accelerazioni, tempo del percorso ecc. Manca però una cosa molto importante, tutte queste analisi sono completamente inutili nel momento un cui l'esaminato fa un errore nell'eseguire il percorso, ad esempio sbaglia lo slalom oppure esce fuori dalla pista. Vediamo adesso come vengono gestite le segnalazioni di entrambi questi errori.

**Posizioni dei coni.** Prima si definisce il motivo per cui abbiamo a disposizione certi dati. Come specificato alla sezione 2.3, oltre alla percorso fatto dalla moto, le altre informazioni che ci fornisce il sistema di rilevamento degli oggetti sono le posizioni iniziali dei coni, e tutti gli eventi che potrebbero avvenire in seguito. Le posizioni dei coni che otteniamo grazie al sistema di rilevamento oggetti sono delle coordinate nello spazio immagine che devono essere proiettate, così come abbiamo fatto per il percorso della moto, utilizzando la matrice di omografia (il cui ottenimento è analizzato in sezione 5.1). Per farlo si utilizza lo stesso metodo che abbiamo usato per proiettare le posizioni della moto (vedere sempre sezione 5.1) ottenendo così le posizioni dei coni nello spazio proiettato, che è proprio quello che ci serve per determinare se l'esaminato ha sbagliato lo slalom oppure è uscito dalla pista.

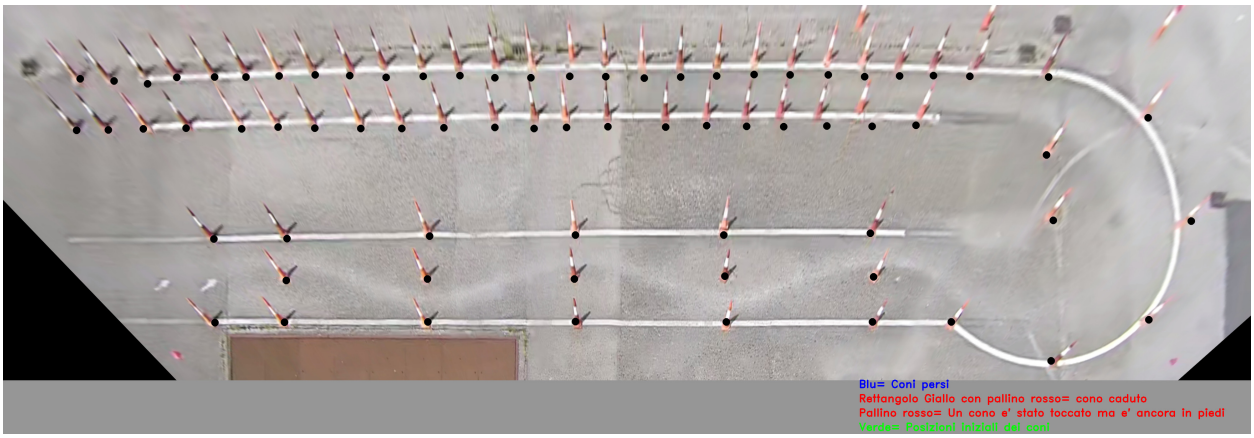



Figura 5.13: In figura è possibile visualizzare le posizioni dei coni post applicazione della proiezione, rappresentate con dei punti neri.

**Controllo sullo slalom e uscita dal percorso.** Adesso che abbiamo definito da dove arrivano le informazioni riguardanti le posizioni dei coni, possiamo parlare del sistema utilizzato per verificare se lo slalom sia eseguito correttamente o meno, e per farlo prendiamo come riferimento l'immagine 5.14. Come è possibile vedere in figura la zona dello slalom è stata divisa in 9 sezioni; ognuna delle sezioni è delimitata da quatto coni, e sapendo bene come ogni cono è dotato di un ID, è abbastanza semplice dividere questa zona del percorso in diverse sezioni.

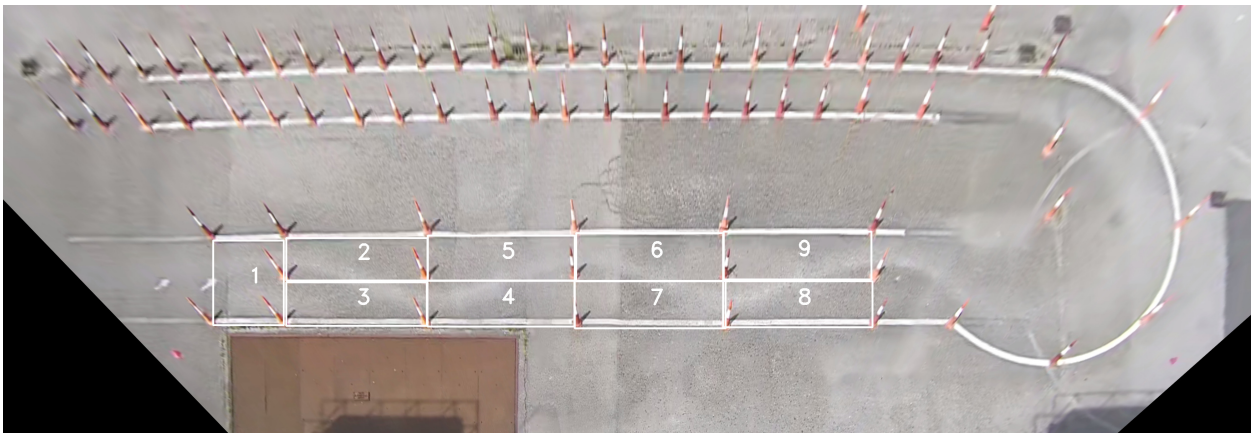



Figura 5.14: In questa figura è possibile vedere come la zona dello slalom nella pista LSB sia stata divisa in 9 sezioni, che ci permetteranno di decidere se lo slalom è stato fatto correttamente oppure no.

Però, per controllare anche che la moto non sia uscita dai limiti del percorso, generalizziamo l'idea di dividere lo slalom in sezioni, dividendo invece l'intera pista in sezioni, in questo modo possiamo tenere sotto controllo l'intera situazione. L'unico ordine corretto di visita delle sezioni

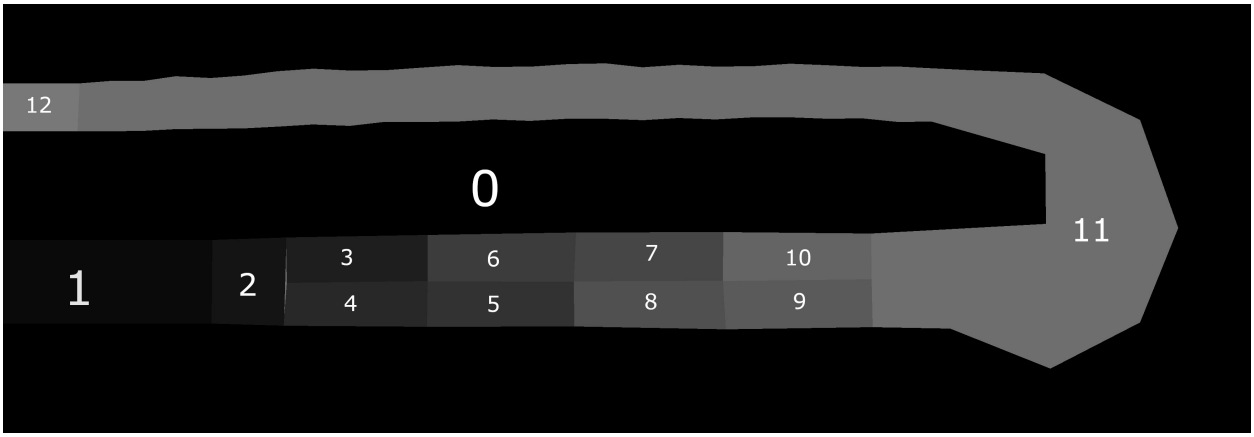



Figura 5.15: In figura è possibile vedere la divisone dell'intera pista in sezioni, da 0 che rappresenta tutto quello che sta fuori dalla pista, a 12 che rappresenta la fine del percorso.

(visibili in figura 5.15) che rappresenta una guida eseguita correttamente è il seguente: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12$. Qualsiasi altro ordine di visita delle sezioni rappresenta una guida eseguita scorrettamente e quindi da punire con una penalità.

L'algoritmo che utilizziamo per controllare che la guida venga eseguita correttamente, utilizza come base proprio l'immagine che vediamo in figura 5.15. Essa non è altro che una matrice, dove ogni cella contiene 3 canali ed ogni canale è un intero di 8bit. Praticamente è una matrice di pixel, ed ogni pixel ha un suo colore definito dal sistema RGB. In questo modo, abbiamo la possibilità di dividere la pista in sezioni e ad ogni sezione assegnare un colore che la contraddistingue, ad esempio per tutti i pixel che si trovano nella zona 1 il loro valore sarà $(1, 1, 1)$ che rappresenta il

colore quasi nero, per tutti i pixel della zona 2 sarà $(2, 2, 2)$ e così via per tutte le altre zone. Detto questo vediamo qual'è l'algoritmo che ci permette di determinare se la guida è stata eseguita correttamente oppure no.

**Algorithm 6** Analisi della traccia

```
1: procedure OutOfTrack(percorsoMoto3D : vettore di punti 3D, lookUpTable : Mat)
2:     firstPoint ← Point2f(percorsoMoto3D[0].x, percorsoMoto3D[0].y)
3:     previous_pixel ← lookUpTable[firstPoint.x][firstPoint.y] : colore RGB (r,g,b)
4:     for i ← 1 to percorsoMoto3D.size() − 1 do
5:         x ← percorsoMoto3D[i].x
6:         y ← percorsoMoto3D[i].y
7:         pixel ← lookUpTable[y : riga][x : colonna]  ▷ Accedo alla matrice come se fosse una
   lookUpTable recuperando il colore associato
8:         if pixel[0] == 0 then
9:             fuoripista ← true
10:            break
11:        end if
12:        if pixel[0] == previous_pixel[0] or pixel[0] − 1 == previous_pixel[0] then
13:            previous_pixel ← pixel
14:            if pixel[0] == 12 then
15:                return "CorrectPath" : string
16:            end if
17:        else
18:            badSlalom ← true
19:            break
20:        end if
21:    end for
22:    if badSlalom then
23:        return "badSlalom" : string
24:    else if fuoripista then
25:        return "outoftrack" : string
26:    else
27:        return "CorrectPath" : string
28:    end if
29: end procedure
```

Per capire meglio come funziona questo algoritmo vediamo di fare un esempio. Ogni percorso deve necessariamente cominciare nella sezione 1. A questo punto iniziamo a iterare su tutte le posizioni del percorso che saranno dei punti del tipo $(x, y)$; dato un punto utilizziamo x e y rispettivamente come indici di colonna e riga per accedere alla matrice "lookUpTable" (5.15), ottenendo un colore RGB del tipo $(r, g, b)$. Supponiamo di partire dalla sezione 1 allora il colore

ottenuto sarà proprio $(1, 1, 1)$, detto questo si continua ad iterare ed accedere alla matrice usando il punto, e per ogni punto si controlla:

- se il colore corrispondente alla posizione nella lookUpTable è $(0, 0, 0)$ allora vorrà dire che siamo finiti fuori dalla pista.
- Se il colore corrispondente alla posizione nella lookUpTable è uguale al colore del pixel precedente allora sono ancora nella stessa sezione.
- se il colore corrispondente alla posizione nella lookUpTable sottratto 1 è uguale al colore del pixel precedente allora vuol dire che siamo passati alla sezione successiva.
- Infine, se il colore corrispondente alla posizione nella lookUpTable è $(12, 12, 12)$ allora vuol dire che siamo arrivati alla fine del percorso senza commettere nessun errore.

Grazie a questo algoritmo siamo in grado di determinare se la guida eseguita dall'esaminato presenta errori oppure no. Da precisare che ci fermiamo nel momento in cui il primo errore viene rilevato, sia che si tratti di slalom sbagliato sia che si tratti di fuori pista.

## 5.6 Fusione delle informazioni

Consideriamo la risoluzione $I_x, I_y$ del sensore di una generica telecamera e un oggetto con coordinate $(O_x, O_y)$ nello spazio reale con coordinate da definire. L'immagine acquisita ha una precisione dipendente dalla risoluzione spaziale (oltre alla point spread function) che si riflette nel numero di pixel per pollici in una data area (detta $ppi$, pixel per inch). In generale, possiamo affermare che allontanandoci dalla telecamera la risoluzione spaziale diminuisce: il medesimo oggetto più vicino al punto di ripresa va a occupare un numero maggiore di pixel rispetto a quando è lontano, a parità di risoluzione.
Durante l'operazione di calcolo della posizione spaziale di un oggetto avremo, ovviamente, un errore più piccolo per gli oggetti aventi un $ppi$ più elevato, ovvero, per gli oggetti vicini.
Nel caso di un solo punto di ripresa non vi è scelta sui dati da utilizzare per localizzare l'oggetto, ma nel caso di più punti di ripresa, come nel caso della pista illustrato in figura 5.16, è importante capire da quale visuale si ha l'informazione migliore e utilizzare quest'ultima.

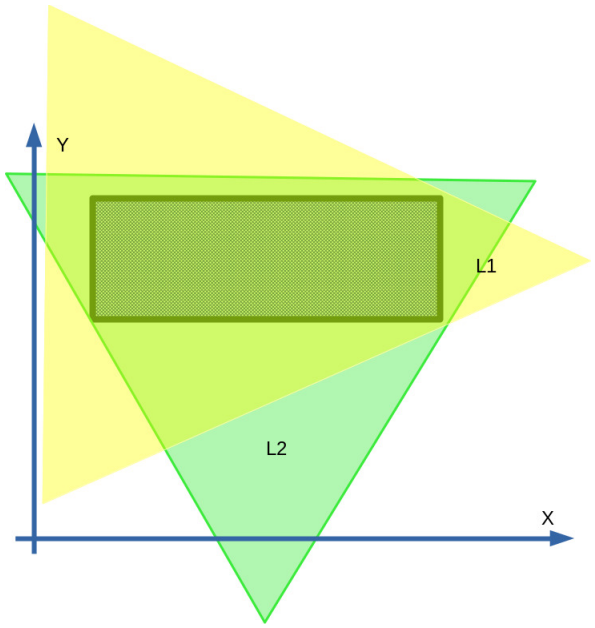



Figura 5.16: schema delle telecamere per la pista LSB.

Come viene calcolata l'informazione migliore? Supponiamo di avere un numero $n$ di telecamere e consideriamo anche i difetti di aberrazione dell'immagine. Indichiamo con l'apice il numero della telecamera. Le coordinate dell'oggetto $(O_x, O_y)$ sono nello spazio reale e corrispondono a coordinate diverse nello spazio del sensore delle telecamere, ovvero $ppi^1_{x(y)}, ppi^1_{y(x)}$ per la prima telecamera, $ppi^2_{x(y)}, ppi^2_{y(x)}$ per la seconda telecamera e $ppi^n_{x(y)}, ppi^n_{y(x)}$ per l'n-esima telecamera. Da notare che i punti $x(y)$ e $y(x)$ per la medesima immagine sono diversi in ogni telecamera in quanto ognuna di esse inquadra la zona da un punto di vista diverso. Identificato il medesimo oggetto dalle diverse inquadrature, è sufficiente utilizzare l'informazione data dalla telecamera che ha una risoluzione maggiore. Nel caso in cui l'immagine sia elaborata da un

programma di riconoscimento automatico (es. della famiglia YOLO 4.1), l'oggetto avente proiezione nello spazio della telecamera con *ppi* maggiore, dovrebbe avere un intervallo di confidenza maggiore (vi sono altri fattori come l'illuminazione che potrebbero invalidare questa ipotesi).


Figura 5.17: Visione della pista da L2.

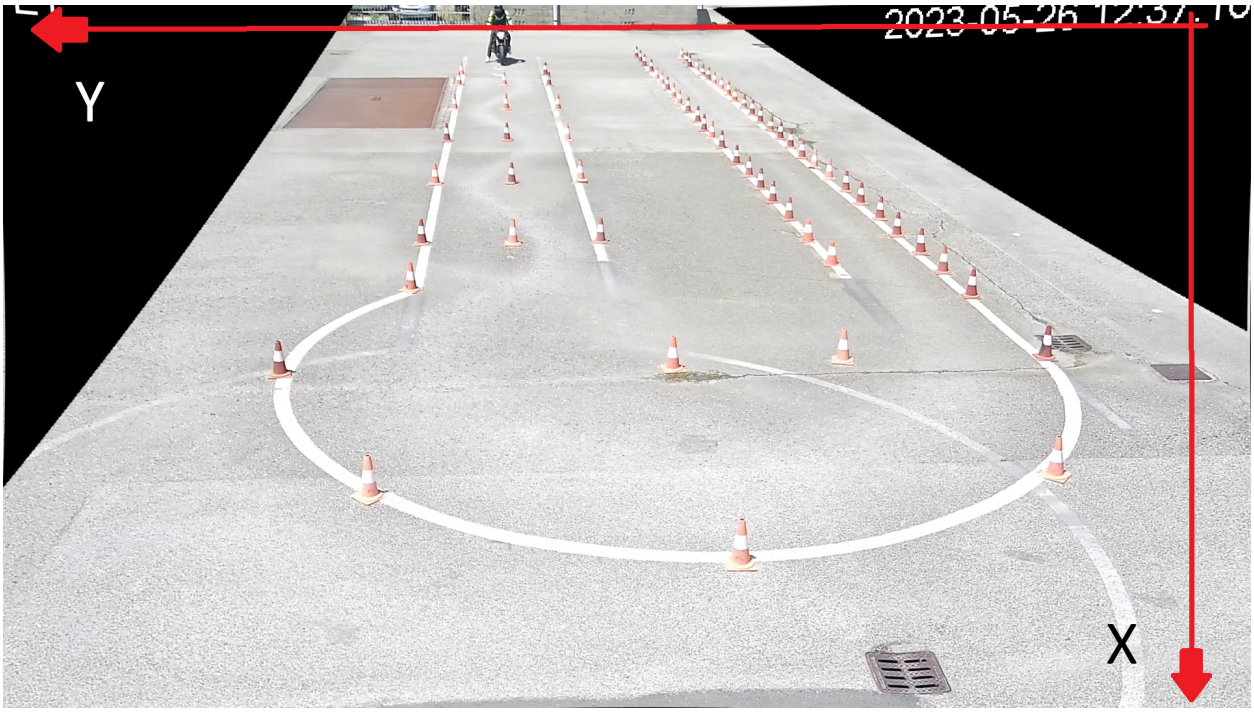



Figura 5.18: Visione della pista da L1.

Nelle immagini in nostro possesso, evidenziate nelle figure 5.17 e 5.18, può essere effettuata una stima qualitativa per ottenere la misura più precisa. Innanzitutto, ovviamente, quando la

visuale dell'oggetto è occlusa per una telecamera, è possibile utilizzare una sola telecamera per il calcolo della posizione e non si pone il problema di scelta.
In relazione alla vista con le due telecamere L1 e L2, la coordinata $x$ sembra avere sempre una maggiore precisione dalla vista della telecamera L2. Per la coordinata $y$ in prima approssimazione può essere sempre utilizzata la telecamera L1 calcolando così il punto come $(O_X, O_y) = f\left(ppi^2_{x(y)}, ppi^1_{y(x)}\right)$, cioè la coordinata $x$ presa sempre dalla telecamera L2, mentre la $y$ da L1.
Dopo un più accurato esame, possiamo osservare però che sulla curva l'ipotesi approssimata non rimane la migliore. Mentre per l'asse $y$ la misura è al più circa 200 pixel da L2 e la misura minima di L1 è di circa 370 pixel, quindi L1 è sempre meglio per la coordinata $y$, per la $x$ la situazione cambia (figure 5.19 e 5.20).

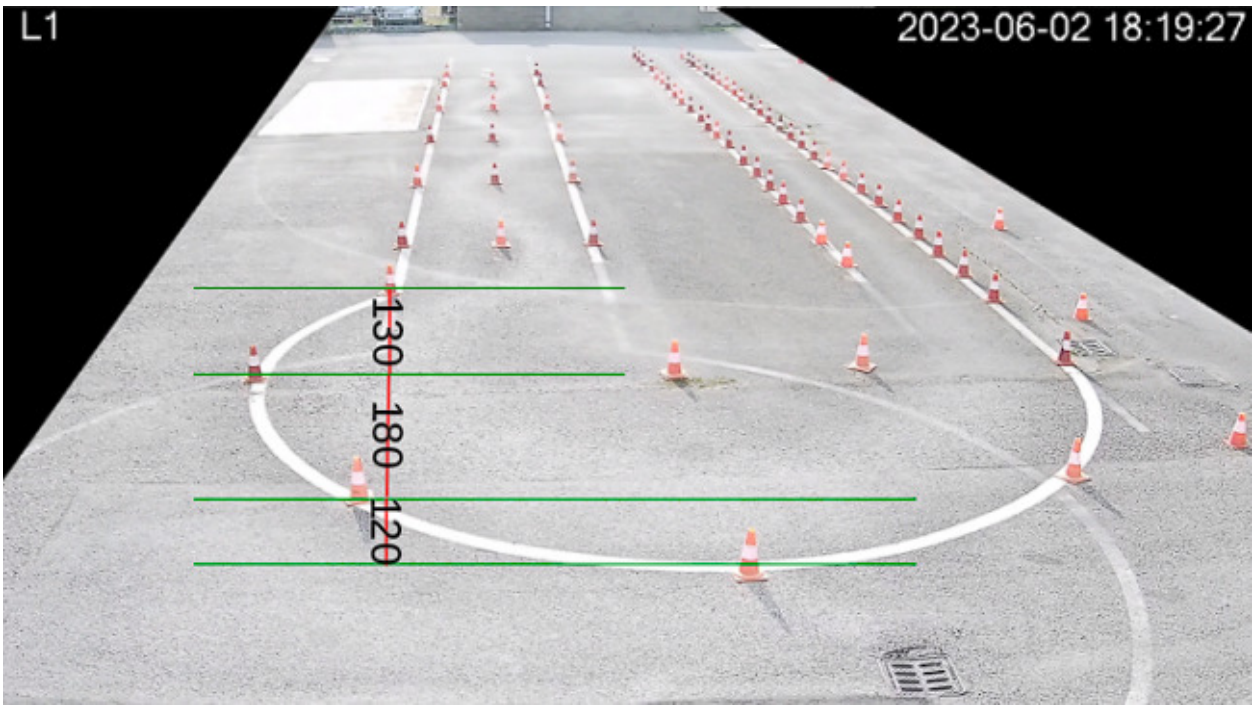



Figura 5.19: Misura in pixel su L1

Sulla curva la telecamera L1 ha una risoluzione migliore per $x$ e per $y$ e quindi dovrebbe essere utilizzata questa per localizzare gli oggetti nello spazio. In generale, la misura migliore da utilizzare è quella della fonte (telecamera) che vede l'oggetto in esame e quelli vicini alla distanza, misurata in pixel, maggiore.

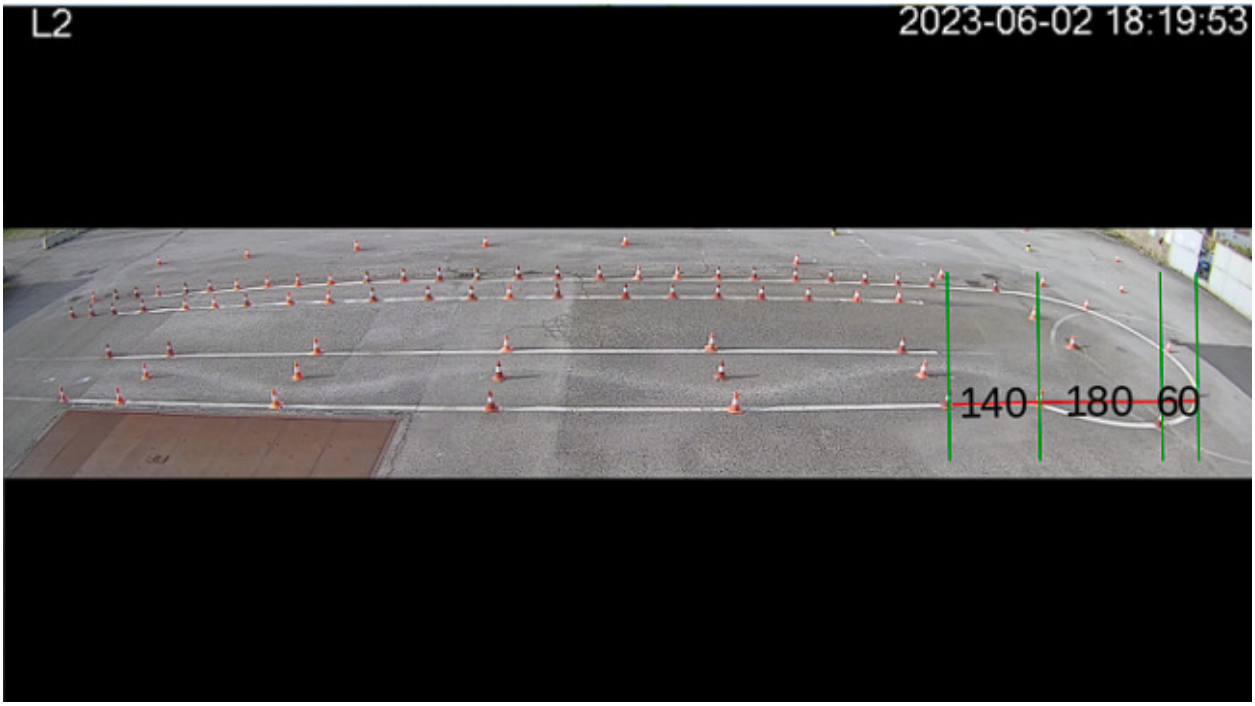



Figura 5.20: Misura in pixel su L2

**Fusione delle traiettorie riprese da L1 e L2.** La fusione delle informazioni riguardo il percorso della moto è un discorso molto delicato, perché dipende tantissimo da quanto i video della telecamera L2 e L1 siano sincronizzati fra loro. Le telecamere per registrare, ricevono un comando via rete ed in seguito cominciano a registrare, quindi a causa di latenze e ritardi nel canale di comunicazione non siamo mai sicuri al $100\%$ che i due video siano sincronizzati. Siccome la moto è in movimento quando viene registrato il video, è necessario che questi siano il più sincronizzati possibile, per avere la possibilità di fondere le due tracce che otteniamo dalla telecamera L2 e L1. Abbiamo così considerato diverse soluzioni:

1. sfruttare il riferimento temporale inserito nel video dalla telecamera per ottenere una sincronizzazione approssimata, ma c'è il rischio che i video siano desincronizzati di almeno un secondo.

2. Usare un SMPTE timecode (vedere sezione 3.7), in modo tale da assegnare ad ogni frame una sequenza numerica univoca che rappresenta l'ora, minuto, secondo in cui il frame è stato registrato. In questo modo ogni frame verrà assegnato un numero univoco, permettendo una sincronizzazione tra fonti video registrate nello stesso momento pressoché perfetta.

3. La terza soluzione era quella di non fare nulla assumendo che i due video fossero già abbastanza sincronizzati. Dopo alcune verifiche empiriche ci siamo resi conto che in condizioni normali la registrazione dei video iniziava praticamente nello stesso momento, e che quindi alla fine i due video risultassero sincronizzati con un errore dell'ordine di qualche millisecondo.

Purtroppo per mancanza di tempo, e possibilità di testare le tre soluzioni a cui avevamo pensato, alla fine per descrivere il percorso della moto utilizziamo solamente le posizioni ottenute dalla registrazione della telecamera L2, senza applicare nessun tipo di fusione con il percorso registrato dalla telecamera L1.

**Fusione dei dati per il calcolo delle coordinate dei Coni.** La fusione delle informazioni sui coni è più semplice rispetto a quella della moto, perché sostanzialmente i coni sono per la maggior parte del tempo fermi, a meno che vengano toccati, ma in quel caso rappresentano un errore da segnalare. Considerando quanto scritto all'inizio di questa sotto-sezione e assumendo che le posizioni dei coni sono già state proiettate grazie l'omografia, vediamo l'algoritmo che viene utilizzato per fondere le informazioni.
Tutte le informazioni che riguardano i coni di L2 e di L1 (specificate in sezione 2.3) sono separate in due strutture dati diverse. La struttura dati scelta è una hashMap, che ha come chiave un intero che è l'ID del cono, e come valore una lista di elementi, ogni elemento rappresenta un evento del cono con delle informazioni che sono: il frame in cui si verifica l'evento, lo stato del cono per quell'evento, è la nuova posizione nello spazio del cono relativa all'evento segnalato.
Come già scritto alla sezione 2.3 un cono può trovarsi in 4 stati diversi:

- 1 è stato iniziale del cono. Definisce se esso è presente oppure no, se un cono non ha stato 1 allora vorrà dire che non è presente per tutto la durata del percorso, quindi non va considerato.
- Lo stato 3 sta a significare che nel frame segnalato il cono con l'ID assegnato non è stato avvistato.
- Stato 2 o 4 sta a significare il cono è stato visto rispettivamente mosso (in posizione diversa da quella iniziale ma ancora in piedi) oppure caduto (posizione diversa da quella iniziale ma capovolto) nel frame segnalato.

L'algoritmo di fusione delle informazioni si basa sul far confluire i dati di entrambe le strutture dati appena descritte, una per L1 e una per L2, in un unica hashMap indicizzata con l'ID del cono, e per ogni ID un unica struct composta dalle seguenti informazioni:

- $frame$ a cui sono relative le informazioni.
- $stato$, $statoL1$ e $statoL2$ che rappresentano rispettivamente lo stato del cono dopo la fusione delle informazioni che abbiamo sugli stati di L1 e L2 (fatta con una politica di decisione dello stato che descriveremo in seguito), lo stato del cono in L1 e lo stato del cono in L2.
- $pos$ che rappresenta la posizione del cono dopo la fusione, secondo la politica stabilita all'inizio di questa sezione, cioè in generale si usa sempre la coordinata $y$ della telecamera L1, però nella curva ed in particolare nei coni con ID dal 19 al 25 utilizziamo sia la coordinata $x$ che quella $y$ della telecamera L1.

In figura 5.21 si osservi il risultato dell'algoritmo di fusione appena descritto. Con i punti in nero vengono rappresentate le posizioni post fusione delle informazioni, invece con dei cerchi vuoti in ciano rappresentiamo le posizioni riprese dalla telecamera L2, mentre in magenta quelle riprese dalla telecamera L1. Si può notare come in tutta la pista usiamo sempre la coordinata $x$ della telecamera L2 (il pallino nero sia sempre alla stessa altezza del cerchio ciano), mentre nella curva ed in particolare nei coni con ID dal 19 al 25, il pallino nero si allinea completamente con il cerchio magenta.

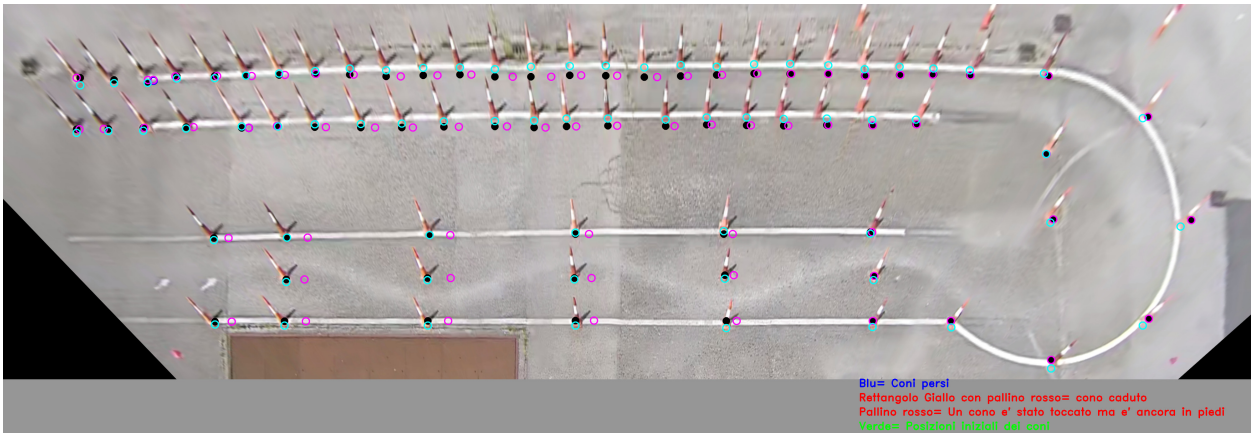



Figura 5.21: In figura è possibile apprezzare il risultato delle fusione delle informazioni sui coni seguendo la politica scelta e descritta in questa sezione.

Per quanto riguarda invece la politica di fusione degli stati del cono, durante la fusione delle informazioni, diciamo che prendiamo come riferimento la telecamera L2 che riesce a riconoscere meglio lo stato in cui si trova il cono, le regole che seguiamo sono le seguenti:

- se gli stati segnalati da L1 e L2 sono uguali allora non c'è nessuna scelta da fare.
- se uno dei due stati è 4 (cono caduto), lo stato fusione sarà proprio 4.
- se lo stato segnalato da L1 è 3 (cono non avvistato), allora confederiamo qualunque stato ci venga segnalato da L2, e viceversa.
- Se lo stato segnalato da una delle due telecamere è 1, se l'altra telecamera segnala un 2 o un 4, allora lo stato fusione sarà 2 o 4, altrimenti sarà 1.
- Lo stato segnalato da una delle due telecamere è 2, se l'altra telecamera segnala un 4, allora lo stato fusione sarà 4, altrimenti viene mantenuto il 2 come stato fusione.

Riassumendo i brevissime parole quanto appena scritto, praticamente abbiamo definito una gerarchia tra gli stati, che è la seguente: 4 (Cono caduto) $\rightarrow$ 2 (Cono mosso) $\rightarrow$ 1 (stato iniziale) $\rightarrow$ 3 (Cono non rilevato).

## 5.7 Rappresentazione virtuale per mezzo di un motore grafico 3D

Nonostante si riescano a riconoscere la maggior parte delle penalità previste con il sistema sviluppato a codice, si è pensato che sarebbe stato interessate avere un riscontro visivo di quello che avveniva su pista, oltre alle immagini mostrate lungo tutto questo capitolo nella quale si poteva osservare la traccia della moto e cosa accadeva ai coni.
Si è deciso quindi di costruire un Digital Twin del percorso, ed in particolare della pista LSB, osservabile in figura 5.22. La ricostruzione è avvenuta seguendo fedelmente le dimensioni specificate dal decreto del Governo [5].

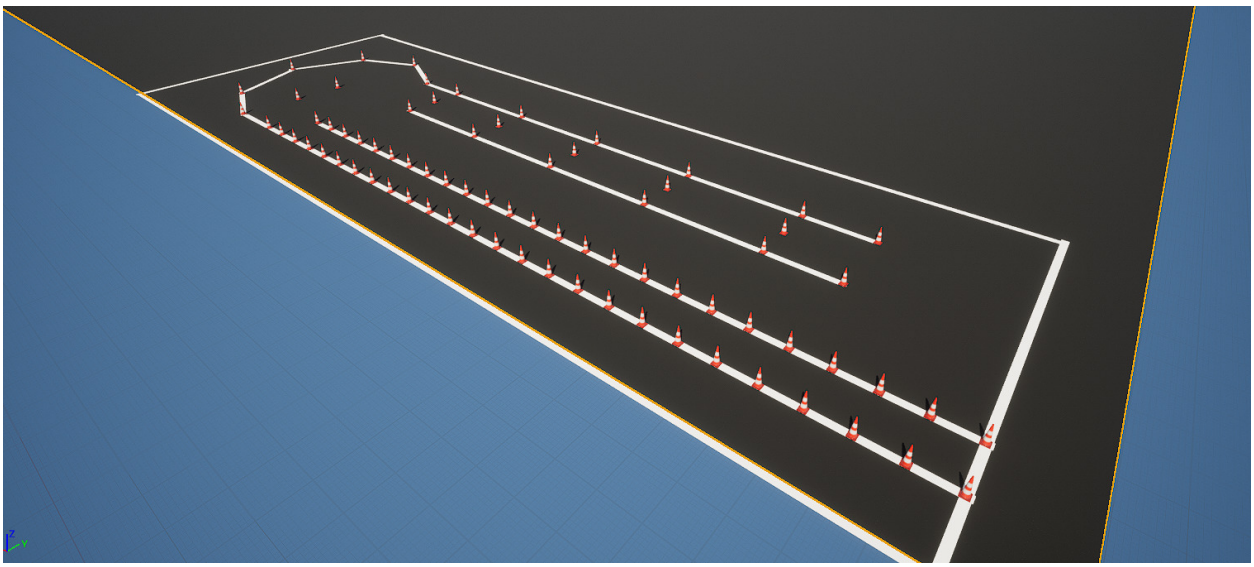


Figura 5.22: Ricostruzione della pista LSB, secondo le misure specificate nel decreto [5] del Governo, in materia di esame di patente.

Nelle figure 5.24 e 5.23 si osservano i modelli 3D usati per rappresentare il cono e la moto. Entrambi sono stati scaricati dal sito ”TurboSquid.com” ([40]), nella quale è possibile recuperare diversi modelli 3D di vari oggetti gratuitamente ma anche a pagamento.

Figura 5.23: Modello 3D del cono usato per modellare la pista.

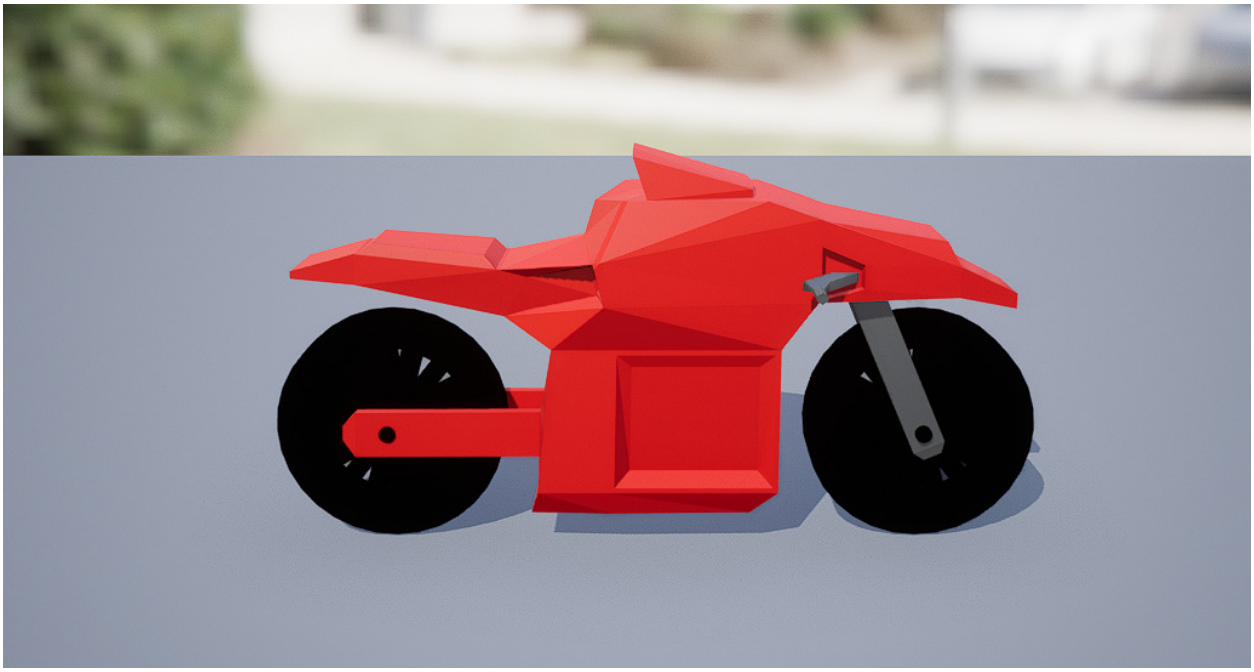


Figura 5.24: Modello 3D della moto.

I sistemi di programmazione messi a disposizione da Unreal Engine 5 per sviluppare un qualcosa sono:

- un sistema di scripting via codice in C++;
- un sistema di scripting visivo basato sui nodi, chiamato Blueprint.

Entrambi i sistemi sono stati progettati per essere utilizzati insieme oppure singolarmente, in base alle necessità. Si decide caso per caso se utilizzare i nodi di Blueprints per sviluppare una

componente del progetto, oppure se usare del codice.
Blueprint non è un linguaggio proprio distinto da C++; viene eseguito su una macchina virtuale- un processo separato che traduce i Blueprint in codice C++. Si dice che Blueprint è un metodo di programmazione C++ basato su interfaccia in cui si utilizzano i nodi per creare elementi di gameplay con l'editor di sistema. Con modifica e programmazione basata sui nodi, si può rapidamente sviluppare codice senza passare per un IDE, consentendo di processare le idee velocemente e solo dopo convertire i Blueprint in codice C++. Quando si lavora utilizzando entrambi i sistemi, è possibile esporre funzioni C++, eventi o variabili ai Blueprint, permettendo anche ai non programmatori di usufruire delle funzionalità di C++ (si faccia riferimento a [41]).
In sostanza Blueprint non è un linguaggio di programmazione isolato per Unreal Engine, ma è un modo per consentire ai non programmatori di programmare, e a tutti i membri del team di estendere il codice C++ scritto. Con C++ si ottiene un accesso più ampio e profondo alle funzionalità principali di Unreal Engine. Si possono anche cambiare o modificare le caratteristiche e le funzionalità principali del motore e dell'editor stesso. Avendo accesso diretto al codice, si ha la possibilità di scrivere funzionalità computazionalmente pesanti o veloci grazie ad un accesso più profondo alle funzioni del motore grafico.
Sebbene un intero progetto possa essere creato utilizzando solamente Blueprint o C++, l'utilizzo combinato dei due sistemi migliorerà di molto le funzionalità fornite.

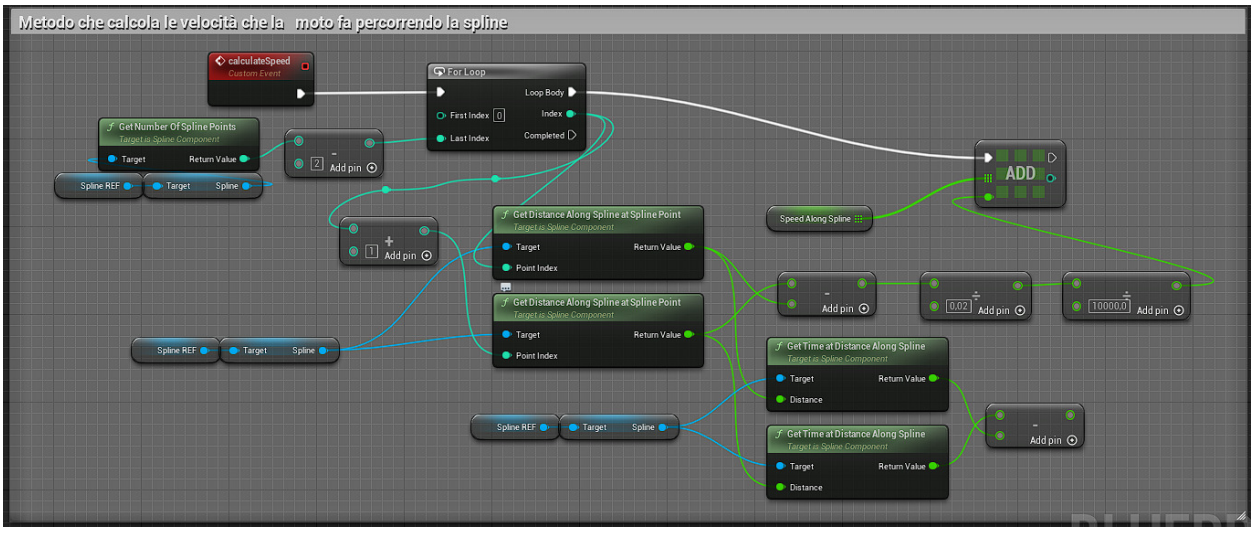



Figura 5.25: Esempio di un metodo sviluppato su Unreal Engine 5, sviluppato con scripting visivo, e che consiste nel calcolare la velocità della moto lungo il percorso ([42])

La traiettoria che si utilizza, sotto forma di file .txt con le posizioni della moto al suo interno, è quella ottenuta dopo i passaggi specificati nelle sezione precedenti, all'interno di questo capitolo, di cui si osserva un esempio della traiettoria in figura 5.11.

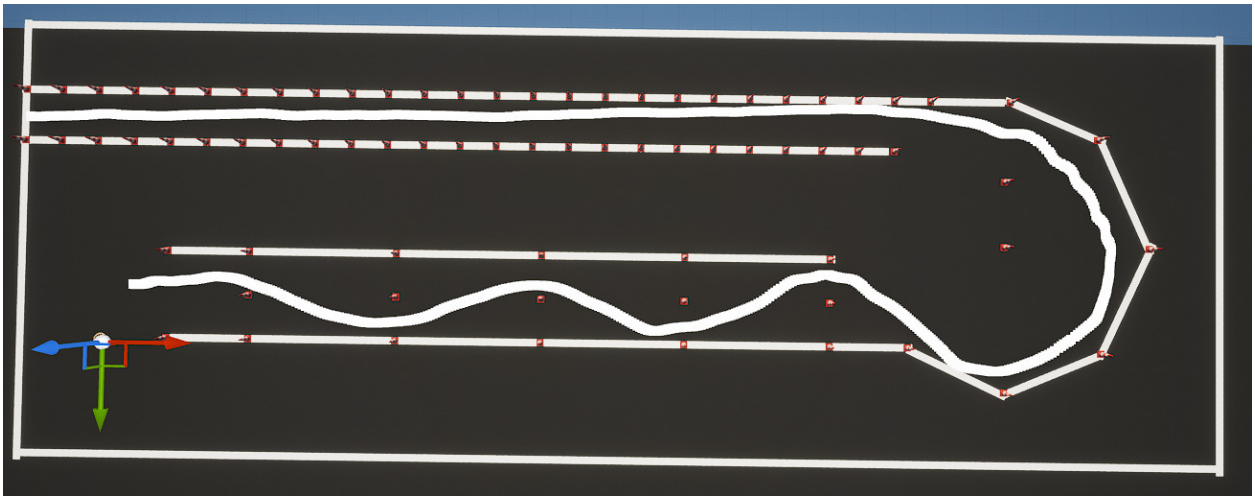


Figura 5.26: In bianco la traiettoria percorsa dalla moto nella ricostruzione virtuale della pista LSB.

In figura 5.26 si osservi quanto ottenuto dalla ricostruzione della pista LSB e dall'utilizzo della traiettoria riferita in precedenza (linea bianca che attraversa la pista).
Il controllo di correttezza del traiettoria viene attuato tramite degli elementi direttamente forniti da Unreal chiamati **Trigger- Box**. ([42]). Come è possibile notare in figura 5.27 l'intera pista è ricoperta di questi elementi, posizionati strategicamente, utilizzati per verificare se il guidatore commette una penalità durante la prova. La "Trigger-Box" è uno dei cosiddetti **Trigger Actors**; i Trigger sono attori utilizzati per provocare il verificarsi di un evento quando interagiscono con altri oggetti nel livello. In altre parole, vengono utilizzati per attivare eventi in risposta a qualche altra azione nel livello. Tutti i Trigger predefiniti sono generalmente uguali, differiscono solo nella forma dell'area di influenza - box, capsula e sfera - utilizzata dal Trigger per rilevare se un altro oggetto lo ha attivato.

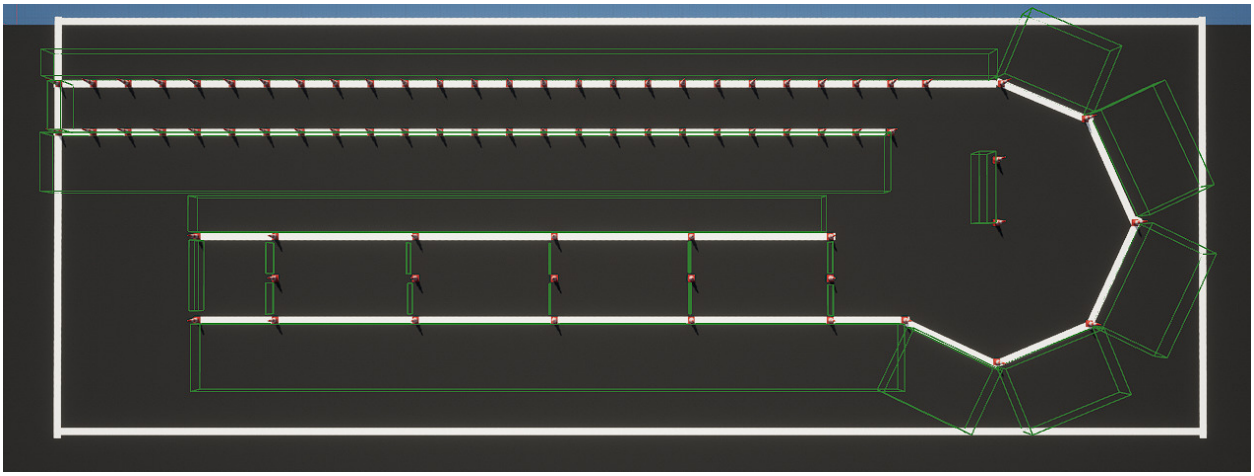


Figura 5.27: In verde si osservino i lati delle figure solide che delimitano le Trigger Box. Si noti come sono posizionate in modo tale da coprire l'intera pista, e rilevare la maggior parte delle penalità richieste.

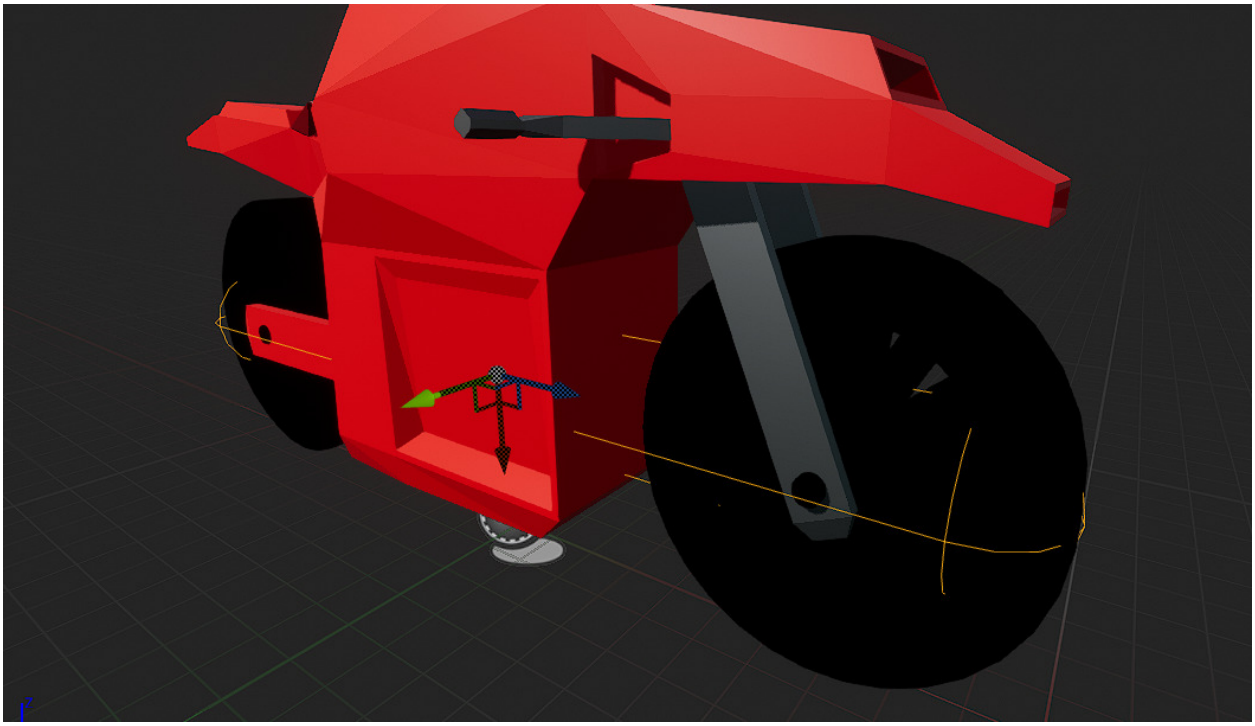


Figura 5.28: Si osservi la capsula che ricopre una parte della moto, ed è una **Trigger-Capsule**.

Per comprendere il sistema dietro il funzionamento della segnalazione di una penalità, si osservi l'immagine 5.28. Si può notare come la zona relative alle ruote della moto è ricoperta con una cosiddetta Trigger Cpasule, la cui grandezza è regolabile tramite l'editor. La logica grazie alla quale andiamo a segnalare se la moto è uscita fuori pista è il seguente: se la Trigger Capsule entra in contatto con una delle Trigger-box (visibili in figura 5.27) che delimitano la pista si mette in

pausa la simulazione e si segnala la penalità corrispondente.
Per il controllo sullo slalom, si utilizza lo stesso sistema, solamente che in questo caso si utilizza della logica per determinare se la moto sta seguendo il percorso corretto oppure no, sempre grazie al passaggio attraverso le Trigger-box" segnalando un errore se la moto passa dalla Trigger-Box sbagliata al momento sbagliato.
Grazie alle Trigger-Box poste all'inizio e alla fine del percorso si va a calcolare il tempo che la moto impiega per percorrerlo. Nel momento in cui la moto entra in contatto con la Trigger-Box iniziale parte un cronometro e quando la stessa entra in contatto con la "Trigger-Box" finale il cronometro si ferma segnalando il tempo finale.
Si prevede anche un sistema che in base alle informazioni ottenuto dai coni, dopo averne fatto la fusione, segnala la caduta o spostamento del relativo cono, cercando di simulare l'avvenimento al frame corrispondente in cui è stato segnalato l'evento.

## 5.8 Sperimentazione del sistema

In questa sezione si osservano i risultati ottenuti su diverse tipologie di percorso, mostrando la capacità del sistema di misurare le grandezze in gioco e di individuare le penalità richieste. Prima di procedere con gli esempi, alcune osservazioni generali:

1. Il prototipo funzionante utilizzato per questi primi risultati riguarda la pista LSB. Tutto il tracciamento della moto è fatto usando solo la telecamera L2, che ha una visuale migliore di tutta la pista. Le informazioni ottenute dalla telecamera L1 sono usate per ottenere una migliore stima della posizione dei coni.

2. Tutte le immagini che si mostreranno in questa sezione sono una vista dall'alto della pista LSB dove i pallini verdi rappresentano le posizioni dei coni all'inizio del test.

3. Come specificato nella sezione 5.4 è previsto il calcolo di un punteggio finale ottenuto dal confronto del percorso che si sta giudicando con un percorso di riferimento effettuato dall'istruttore. Per una definizione accurata dei coefficienti della media pesata della formula 5.16 sono necessari molti più dati di quelli a nostra disposizione; si è preferito dunque mostrare solo i singoli parametri misurati lasciando libero il campo punteggio nelle tabelle di valutazione del percorso.

4. Il sistema è in grado di stimare le accelerazioni e le grandezze derivate da essa, ma, a causa della eccessiva rumorosità dei dati raccolti, le quantità calcolate sono risultate poco rappresentative delle caratteristiche del percorso. Per questo motivo i punteggi calcolabili su queste grandezze non stati inclusi nelle sperimentazione che si mostreranno in questa sezione.

**Percorso di riferimento.** In questo paragrafo si mostrano visivamente tutte le fasi di elaborazione che vengono effettuate per giudicare un test di guida. Utilizziamo come esempio i dati relativi al percorso di riferimento, quello effettuato dall'istruttore della scuolaguida. Alla fine si confrontano i dati il percorso con se stesso per verificare che tutti gli indici siano unitari (massimo punteggio raggiungibile).


Figura 5.29: Traccia della moto registrata dalla telecamera L2.

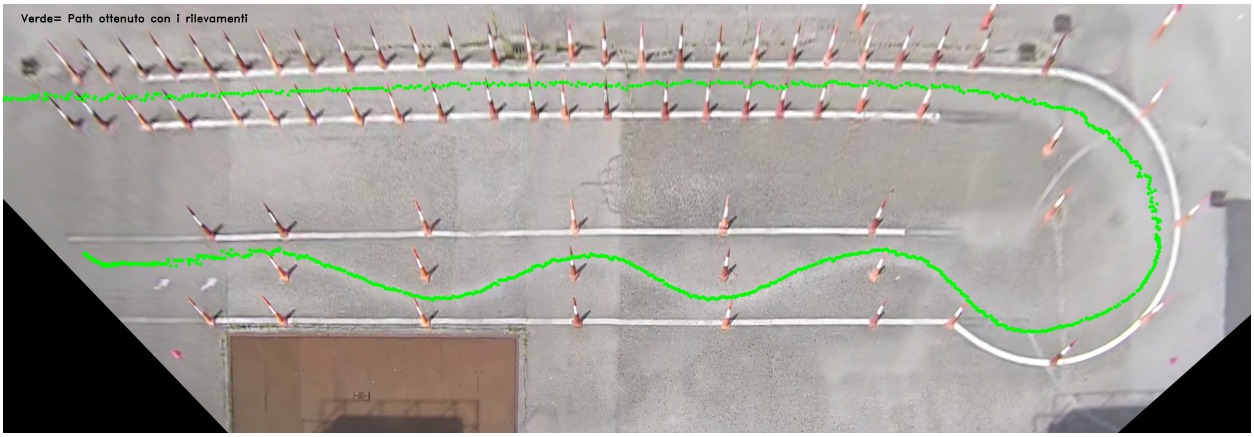



Figura 5.30: Traccia della moto post applicazione dell'Omografia.

Nelle due immagini precedente si mostra come la traccia della moto, dopo essere ripresa dalla telecamere L2, venga proiettata sullo spazio delle coordinate reali, attraverso l'Omografia, produ-

cendo così la seconda immagine. In entrambe, con dei pallini verdi si va a rappresentare la traccia della moto, dove ogni singolo pallino rappresenta la posizione della stessa in un frame specifico.

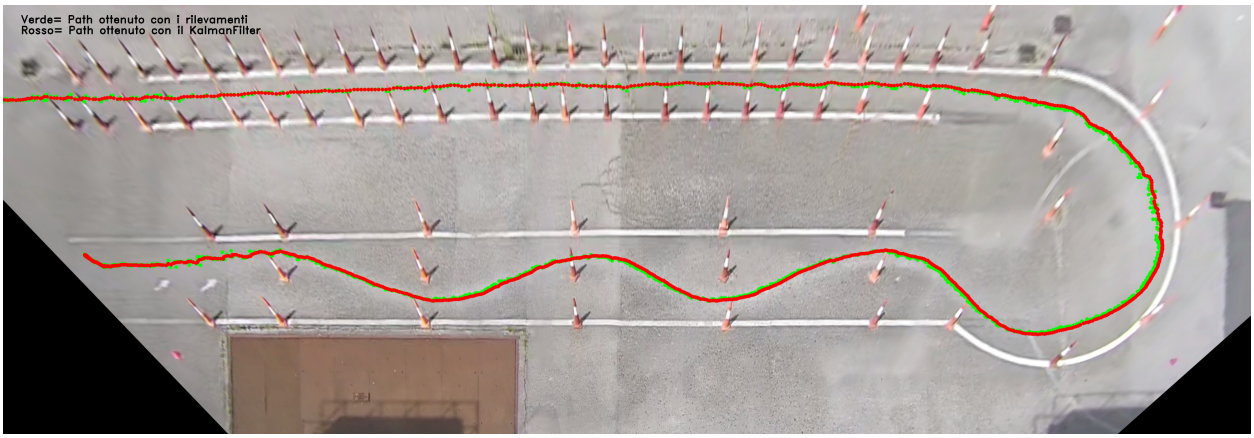



Figura 5.31: In rosso si rappresentata la traccia della moto dopo l'applicazione del filtro di Kalman.

Nell'immagine precedente si mostra il risultato dell'applicazione del filtro di Kalman. Si può vedere come in rosso venga rappresentata la traccia della moto post applicazione del filtro di Kalman, e come questa traccia sia meno rumorosa dell'originale.

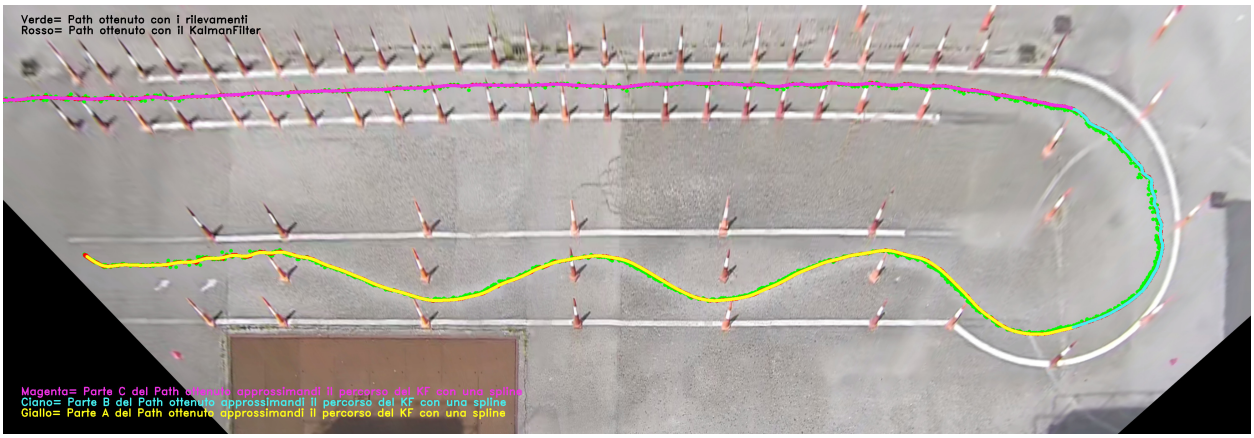



Figura 5.32: Traccia della moto suddivisa in tre zone per la generazione della spline. In giallo c'è la zona A del percorso, in Ciano la zone B, ed in Magenta la zona finale C.

Il motivo per cui utilizziamo le spline è per ridurre il rumore in modo maggiore di quanti dovremmo aver già fatto con l'applicazione del filtro di Kalman. Invece, il motivo della divisione della traccia in tre zone distinte è discusso nella sezione 5.3.

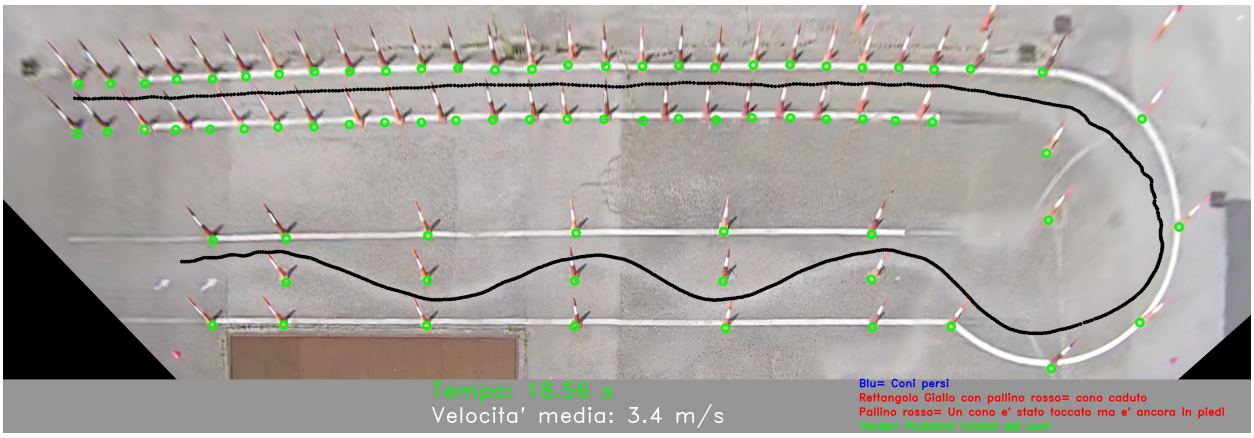



Figura 5.33: Traccia finale della moto, privata delle parti superflue relative all'inizio ed alla fine prima di entrare e dopo essere usciti dalla pista. Nella barra informativa in basso, si indicano le caratteristiche del percorso e si segnalano le eventuali penalità.

In quest'ultima immagine si va a rappresentare la traccia finale percorsa dalla moto, dopo le varie fasi del sistema. Si può notare come con dei punti verdi si rappresenti la posizione del cono all'inizio del test, mentre la traccia nera rappresenta il percorso della moto. La traccia è composta da una serie di punti, ognuno di essi rappresenta una posizione nello spazio ed è collegata ad un frame del video di partenza.
Nella barra informativa in basso si indicano diverse informazioni relative al percorso sostenuto dalla moto, in questo caso velocità media e tempo impiegato. Inoltre è presente una legenda dei simboli che si potrebbero trovare nell'immagine.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 202.368 $m^2$ | 1 |
| Coefficiente correlazione x | | 1 | 1 |
| Coefficiente correlazione y | | 1 | 1 |
| Tempo | 18.5602 s | 18.5602 s | 1 |
| Velocità massima | 571.41 cm/s | 571.41 cm/s | 1 |
| Velocità minima | 27.10 cm/s | 27.10 cm/s | 1 |
| Velocità media | 340.20 cm/s | 340.20 cm/s | 1 |
| Deviazione Standard velocità | 100.36 cm/s | 100.36 cm/s | 1 |
| Punteggio finale | | | |

Tabella 5.1: Punteggi del percorso di riferimento confrontato con se stesso.

Osservando la tabella 5.1, si nota come confrontando il percorso di riferimento con se stesso il punteggi siano tutti uguali a 1.
Si noti la mancanza del punteggio finale, ed il motivo ne è stato dato all'inizio di questa sezione, ed in maniera più specifica alla sezione 5.4.

Da questo paragrafo in poi si assumono come viste tutte le immagini relative a processi intermedi del sistema, e si mostreranno solo i risultati finali, di importanza più rilevante, sulla traccia della moto e sui coni quando necessario.

**Percorso con esito positivo.** In questo paragrafo si confronta un percorso ”buono” con il percorso di riferimento.

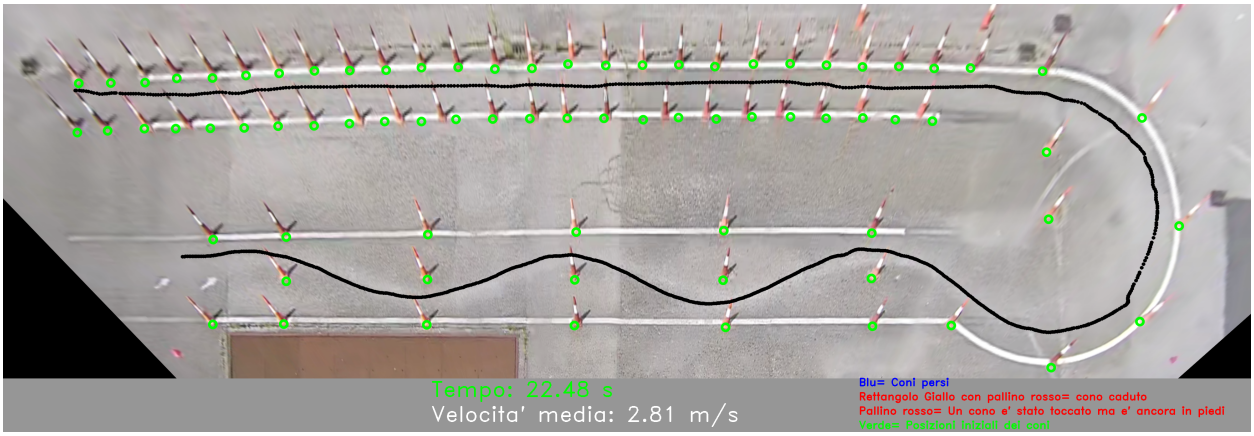



Figura 5.34: Traccia finale della moto nel caso di percorso ”buono”. Il percorso è pressoché identico a quello ideale, e non c'è nulla da segnalare visto che non vi è nessuna penalità. Nella barra informativa in basso si indica tempo di esecuzione e velocità media.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 201.124$m^2$ | 0.993894 |
| Coefficiente correlazione x | | 0.941088 | 0.941088 |
| Coefficiente correlazione y | | 1 | 1 |
| Tempo | 18.5602 s | 22.48s | 0.825621 |
| Velocità massima | 571.41 cm/s | 743.506 cm/s | 0.768539 |
| Velocità minima | 27.10 cm/s | 9.79538 cm/s | 0.610291 |
| Velocità media | 340.20 cm/s | 281.2 cm/s | 0.852187 |
| Deviazione Standard velocità | 100.36 cm/s | 102.899 cm/s | 0.975282 |
| Punteggio finale | | | |

Tabella 5.2: Punteggi del percorso ”buono”, confrontato con il percorso di riferimento.

**Percorso Troppo Veloce.** In questo paragrafo si confronta il percorso si riferimento con un percorso affrontato troppo velocemente, cioè eseguito in meno di 15 secondi (il regolamento specifica che non bisogna scendere sotto tale limite).

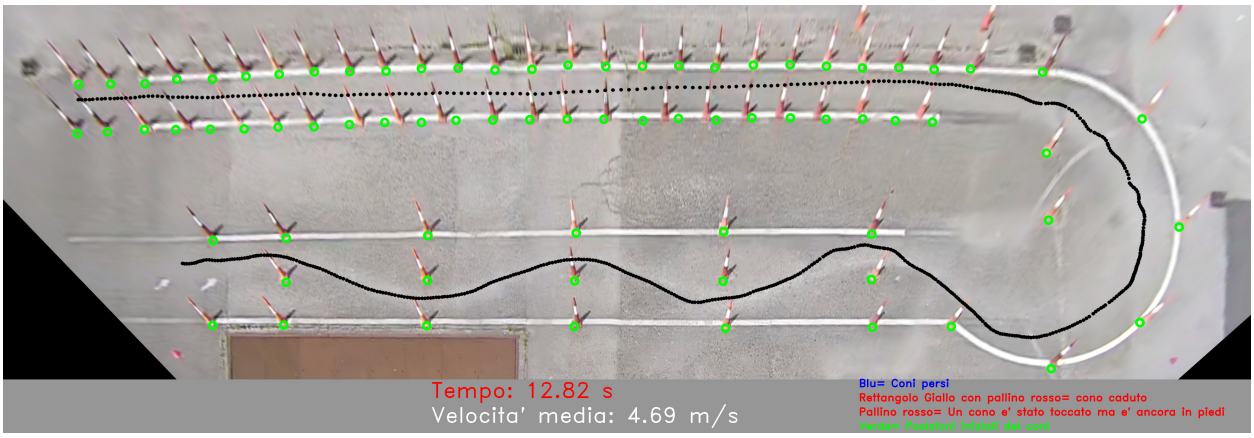



Figura 5.35: Traccia finale della moto nel caso di percorso "troppo veloce". Questa traccia è sostanzialmente molto simile a quella del percorso di riferimento con qualche differenza che riguarda principalmente la distanza tra i punti che la definiscono. Nella barra informativa si osservi come il tempo è indicato in rosso perché inferiore al limite consentito.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 201.401$m^2$ | 0.995244 |
| Coefficiente correlazione x | | 0.995709 | 0.995709 |
| Coefficiente correlazione y | | 0.996716 | 0.996716 |
| Tempo | 18.5602 s | 12.82 s | 0.763784 |
| Velocità massima | 571.41 cm/s | 1158.15 cm/s | 0.493384 |
| Velocità minima | 27.10 cm/s | 65.79 cm/s | 0.411908 |
| Velocità media | 340.20 cm/s | 468.62 cm/s | 0.725976 |
| Deviazione Standard velocità | 100.36 cm/s | 219.88 cm/s | 0.457851 |
| Punteggio finale | | | |

Tabella 5.3: Punteggi del percorso eseguito troppo velocemente, confrontato con il percorso di riferimento.

Osservando la tabella 5.3, si noti il confronto tra il percorso di riferimento e un percorso eseguito troppo velocemente: alcuni punteggi su specifiche caratteristiche del percorso sono più bassi di altri, ad esempio il tempo, oppure i punteggi sulle varie velocità registrate, mentre i punteggi relativi alla forma della traiettoria sono molto alti. Questo è dovuto al fatto che il percorso sostenuto dalla moto è di base simile a quello di riferimento nella traiettoria, però le velocità ed il tempo con cui è affrontato sono totalmente diverse.

**Percorso Lento.** In questo paragrafo si confronta un percorso eseguito lentamente con il percorso di riferimento.

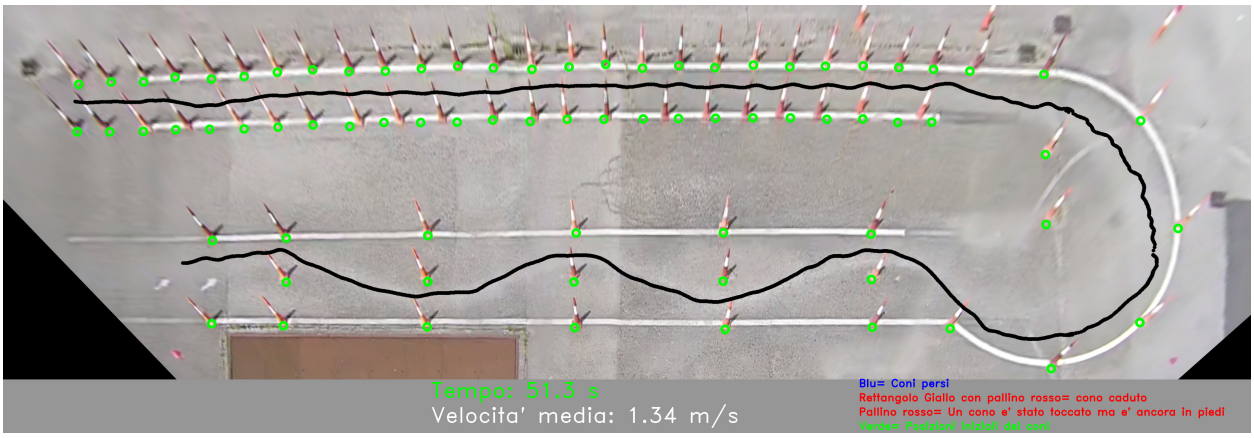



Figura 5.36: Traccia finale della moto nel caso di percorso ”lento”. L’andamento della traccia è molto più titubante e meno deciso. Nonostante il tempo di completamento sia di ben 51 secondi, non è segnato in rosso perché tale evento non costituisce penalità.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 204.845 $m^2$ | 0.987908 |
| Coefficiente correlazione x | | 0.92647 | 0.92647 |
| Coefficiente correlazione y | | 1 | 1 |
| Tempo | 18.5602 s | 51.30 s | 0.361791 |
| Velocità massima | 571.41 cm/s | 649.81 cm/s | 0.879346 |
| Velocità minima | 27.10 cm/s | 21.11 cm/s | 0.818896 |
| Velocità media | 340.20 cm/s | 133.78 cm/s | 0.622372 |
| Deviazione Standard velocità | 100.36 cm/s | 44.06 cm/s | 0.640633 |
| Punteggio finale | | | |

Tabella 5.4: Punteggi del percorso ”lento” confrontato con il percorso di riferimento.

Come si nota dalla tabella 5.4, si confronta il percorso di riferimento con un percorso eseguito lentamente, in cui sono stati impiegati ben 51 secondi per completarlo. Osservando l’immagine 5.36 si può vedere come il percorso ottenuto sia molto traballante rispetto agli esempi visti in precedenza. Si può notare infatti come il punteggio sul tempo sia molto più basso rispetto agli, questo perché c’è una considerevole differenza di secondi spesi nell’eseguire il percorso. Un fenomeno simile si avverte sia per il punteggio sulla velocità media che per la deviazione standard della velocità, ma in minore quantità.

**Percorso in cui la moto si ferma.** In questo paragrafo si mostrano i risultati ottenuti dal confronto del percorso di riferimento con un percorso in cui la moto presumibilmente si ferma, implicando la penalità di ”piede a terra”.

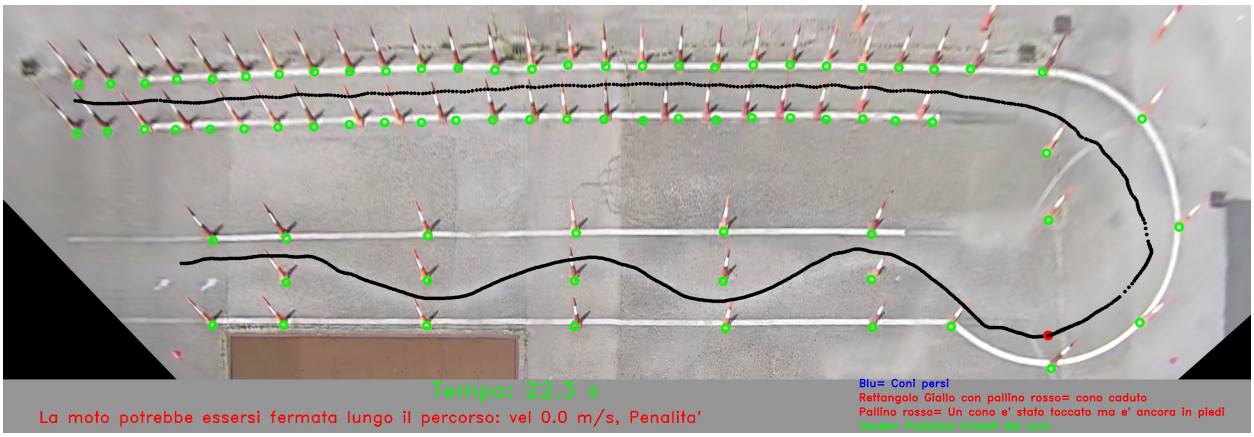



Figura 5.37: Traccia finale della moto nel caso in cui la moto si ferma. Il tempo di completamento viene sempre indicato nella barra informativa, ma si segnala anche che la moto potrebbe essersi fermata lungo il percorso con possibile penalità di ”piede a terra”; in pallino rosso indica la posizione dove si è rilevata la velocità nulla.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 202.594 $m^2$ | 0.998882 |
| Coefficiente correlazione x | | 0.988416 | 0.988416 |
| Coefficiente correlazione y | | 0.984202 | 0.984202 |
| Tempo | 18.5602 s | 22.3003 s | 0.832286 |
| Velocità massima | 571.41 cm/s | 1071.24 cm/s | 0.533413 |
| Velocità minima | 27.10 cm/s | 0.00228882 cm/s | 0.500021 |
| Velocità media | 340.20 cm/s | 230.788 cm/s | 0.756642 |
| Deviazione Standard velocità | 100.36 cm/s | 164.684 cm/s | 0.609385 |
| Punteggio finale | | | |

Tabella 5.5: Punteggi del percorso in cui la moto si ferma, confrontato con il percorso di riferimento.

Osservando la tabella 5.5, si noti il confronto tra il percorso di riferimento ed un percorso in cui la moto si è presumibilmente fermata: nella immagine 5.37 il pallino rosso indica il punto in cui la moto potrebbe essersi fermata. Allo stato attuale del software, questa posizione è calcolata in maniera molto grezza, prendendo semplicemente quella coppia di posizioni in cui viene registrata la velocità estremamente vicina allo zero; si precisa però che ci sono altri modi per calcolare questa posizione, ad esempio una delle idee che avevamo valutato si basava sull'analizzare le velocità lungo il percorso e vedere se in intervalli del percorso, la cui grandezza era da definire, la velocità rimaneva al di sotto di una certa soglia che sarebbe stata molto vicina allo zero.
Un percorso in cui la moto si ferma è strettamente collegato alla penalità ”piede a terra”. Questa penalità è molto difficile da riconoscere: in questo momento siamo solo in grado di riconoscere

se la velocità della moto arriva a zero durante il percorso e quello è un probabile indice di errore. Sfortunatamente, molto spesso c'è un tocco quasi impercettibile con il piede a terra, nel caso in cui si perda l'equilibrio; è necessario lavorare ancora sulla segnalazione di questa penalità magari analizzando la postura del guidatore.

**Percorso con cono toccato.** In questo paragrafo si mostrano i risultati ottenuti dal confronto tra un percorso in cui la moto tocca un cono ed il percorso di riferimento.

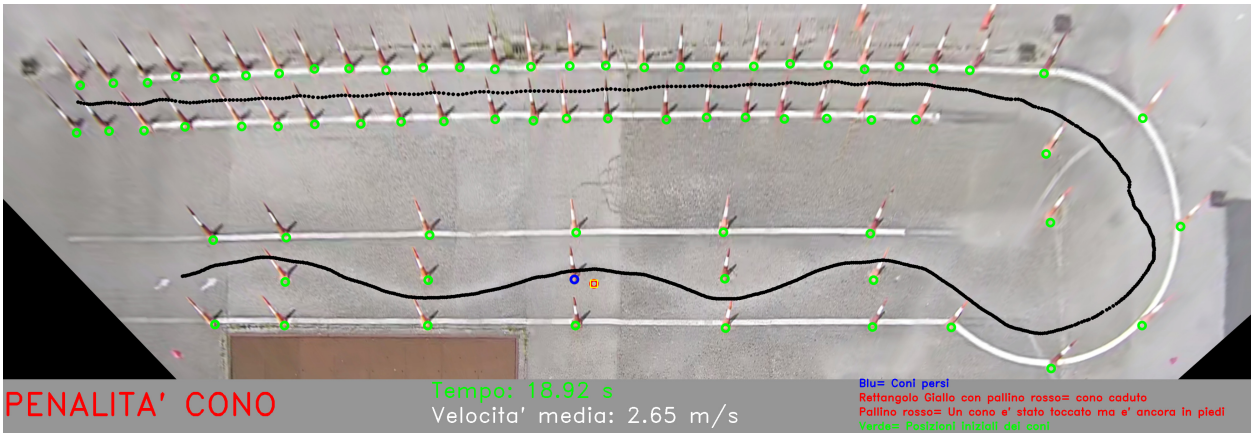



Figura 5.38: Traccia finale della moto nel caso di "cono toccato". Il pallino rosso indica la nuova posizione del cono; se è circoscritto da un quadrato giallo vuol dire che il cono è caduto. Il pallino blu indica la posizione originale del cono osservato e quelli verdi le posizioni di tutti gli altri coni all'inizio del test.

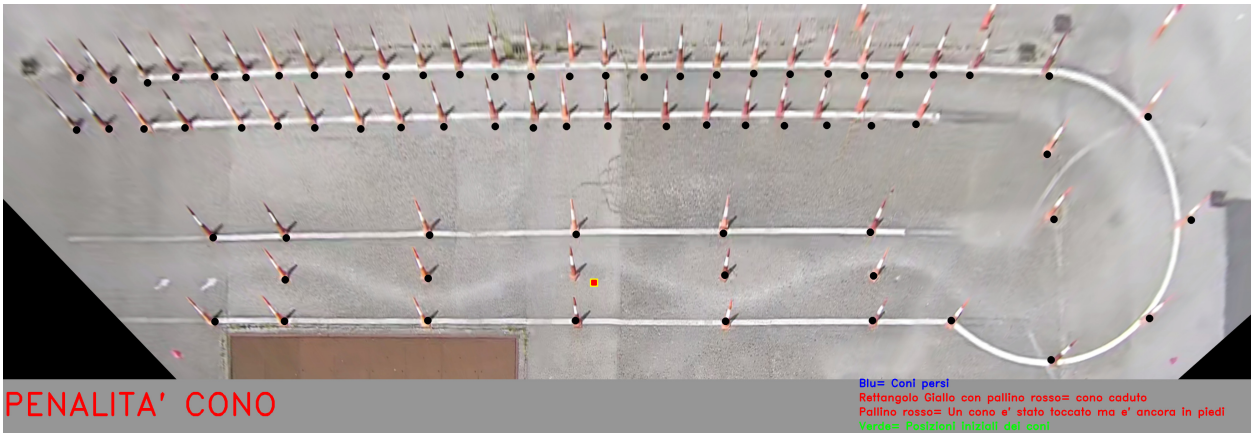



Figura 5.39: Posizioni dei coni dopo aver applicato il processo di fusione delle informazioni ottenute dalle telecamere L1 e L2.

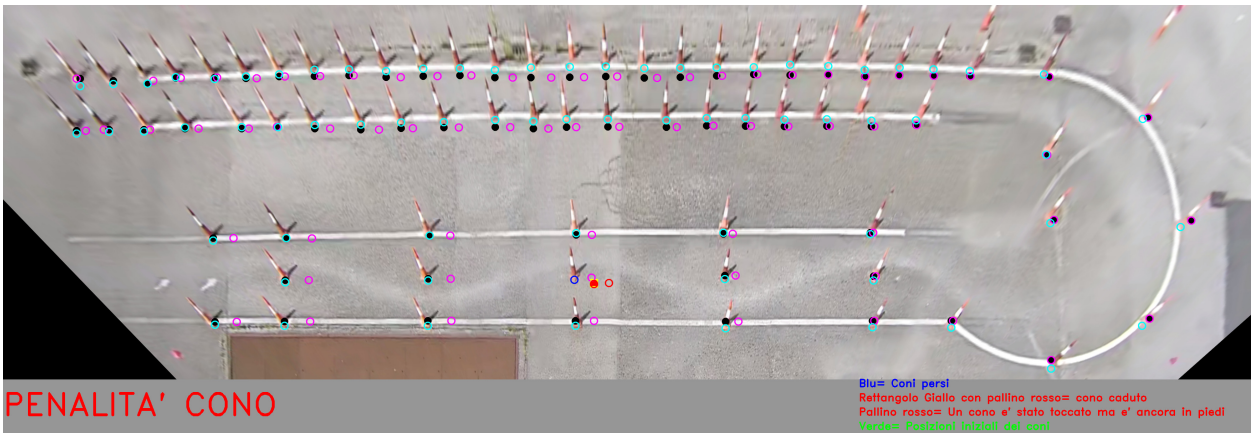



Figura 5.40: Posizioni dei coni ottenute con il processo di fusione, ma anche le rispettive posizioni dei coni prese dalle telecamere L1 (Magenta) e L2 (Ciano).

Osservando le prime due immagini 5.38, 5.39 si noti come con i pallini verdi nella prima immagine si indicano le posizioni dei coni all'inizio del test, con i pallini neri nella seconda si rappresentano le posizioni dei coni dopo la fusione delle informazioni, con il pallino blu si indica un cono perso; ciò non vuol dire che il cono è perso per sempre ma solo che in uno o più frame esso non è stato rilevato; con il pallino rosso circondato dal rettangolo giallo si indica un cono caduto. Si può anche notare come nella zona in cui si trova il cono caduto la traiettoria della moto passa molto vicina al cono in questione a riprova del fatto che c'è un probabilità molto alta che l'evento sia avvenuto.
Analizzando l'immagine 5.40 si può notare invece il risultato del processo di fusione applicato alla posizione dei coni (l'idea dietro è trattata in sezione 5.6). In nero si rappresentano le posizioni dei coni derivanti dalla fusione delle informazioni della telecamera L2 e L1, i cerchi vuoti in Magenta rappresentano le posizioni dei coni ripresi dalla telecamera L1, mentre in Ciano le posizioni riprese dalla telecamera L2. Si può notare come nei coni sulla curva i punti neri combacino alla perfezione con le posizioni riprese dalla camera L1 mentre nel resto della pista si usa sempre la coordinata x della camera L2 e la y di L1, il motivo per cui questo avviene è approfondito in sezione 5.6.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 201.093 mˆ2 | 0.99374 |
| Coefficiente correlazione x | | 0.981505 | 0.973566 |
| Coefficiente correlazione y | | 0.998595 | 0.998595 |
| Tempo | 18.5602 s | 18.9202 s | 0.980972 |
| Velocità massima | 571.41 cm/s | 710.03 cm/s | 0.804774 |
| Velocità minima | 27.10 cm/s | 24.7178 cm/s | 0.919165 |
| Velocità media | 340.20 cm/s | 319.125 cm/s | 0.941642 |
| Deviazione Standard velocità | 100.36 cm/s | 134.73 cm/s | 0.744866 |
| Punteggio finale | | | |

Tabella 5.6: Punteggi del percorso ”cono toccato”, confrontato con il percorso di riferimento.

In tabella 5.6 si possono notare i punteggi ottenuti confrontando il percorso ottimo con un percorso in cui la moto ha toccato un cono. Si può notare come, tralasciando il fatto che il percorso non è valido perché la moto ha toccato un cono, i punteggi relativi alle varie caratteristiche del percorso sono molto alti, proprio in virtù del fatto che non solo la traiettoria ma anche tutte le altre quantità calcolare sono molto vicine a quelle del percorso ottimo con cui viene fatto il confronto.

**Percorso con errore durante la fase di slalom.** In questo paragrafo si mostrano i risultati ottenuti dal confronto di un percorso con ”slalom sbagliato” ed il percorso di riferimento.

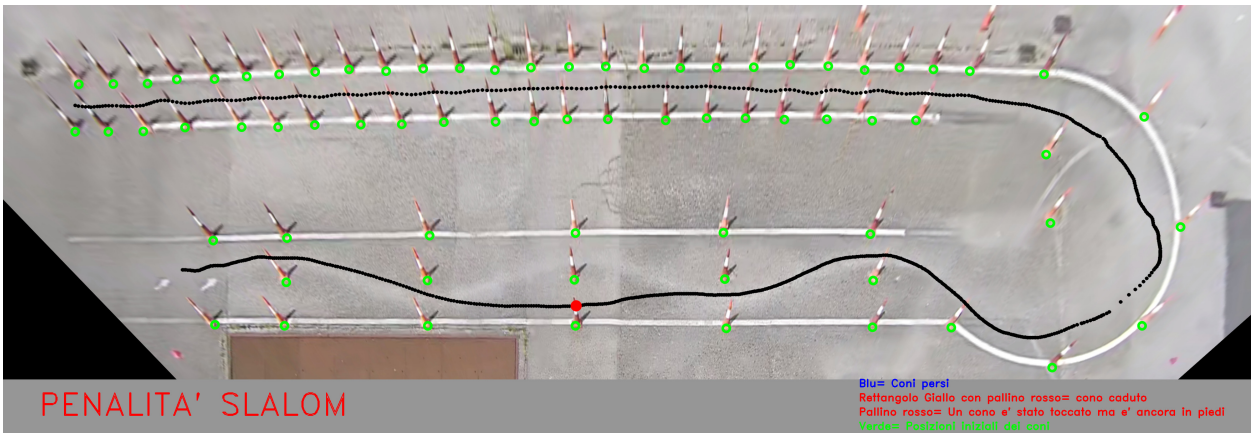



Figura 5.41: Traccia finale della moto nel caso di percorso con ”slalom sbagliato”. Il pallino rosso indica la posizione in cui avviene l’errore relativo allo slalom; nella barra informativa in basso è possibile vedere la penalità segnalata.

Osservando l’immagine 5.41, si noti come viene indicata la posizione in cui lo slalom viene sbagliato con un pallino rosso.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 199.659 $m^2$ | 0.986792 |
| Coefficiente correlazione x | | 0.984222 | 0.984222 |
| Coefficiente correlazione y | | 0.993577 | 0.993577 |
| Tempo | 18.5602 s | 16.9402 s | 0.919722 |
| Velocità massima | 571.41 cm/s | 2100.21 cm/s | 0.272074 |
| Velocità minima | 27.10 cm/s | 11.1735 cm/s | 0.629838 |
| Velocità media | 340.20 cm/s | 347.571 cm/s | 0.978821 |
| Deviazione Standard velocità | 100.36 cm/s | 176.159 cm/s | 0.56969 |
| Punteggio finale | | | |

Tabella 5.7: Punteggi del percorso ”slalom sbagliato”, confrontato con il percorso di riferimento.

Osservando la tabella 5.7, e concentrandosi in particolare sulla voce Velocità massima, si noti come essa sia un velocità non è rappresentativa dell'effettiva traiettoria fatta dalla moto, anzi osservando bene le immagini è anche possibile dedurre da dove arrivi questa velocità di punta così elevata. Questo è un problema dovuto ancora una volta all'eccessiva rumorosità dei dati, nonostante l'utilizzo del filtro di Kalman. Quanto appena scritto, in alcuni casi, ci porta ad avere dei punteggi molto bassi, come avviene per la velocità massima in questa specifica situazione. Nelle immagini successive (fig. 5.42 e 5.43) si mostrano altri esempi di percorso con slalom sbagliato.

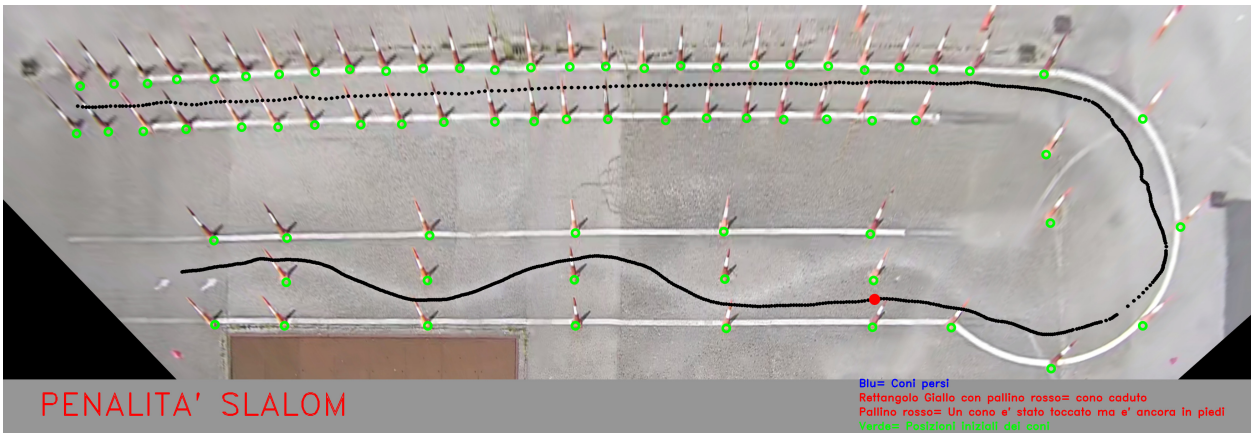



Figura 5.42: Altro esempio di riconoscimento ”slalom sbagliato”.

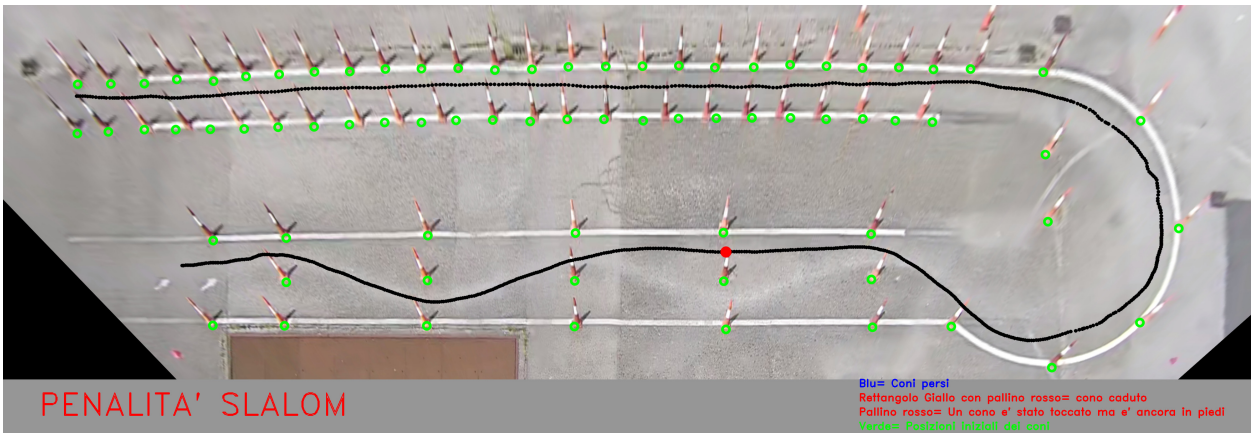



Figura 5.43: Altro esempio di riconoscimento ”slalom sbagliato”.

**Percorso con uscita dalla pista.** In questo paragrafo si mostrano i risultati ottenuti dal confronto tra un percorso con uscita dalla pista ed il percorso di riferimento.

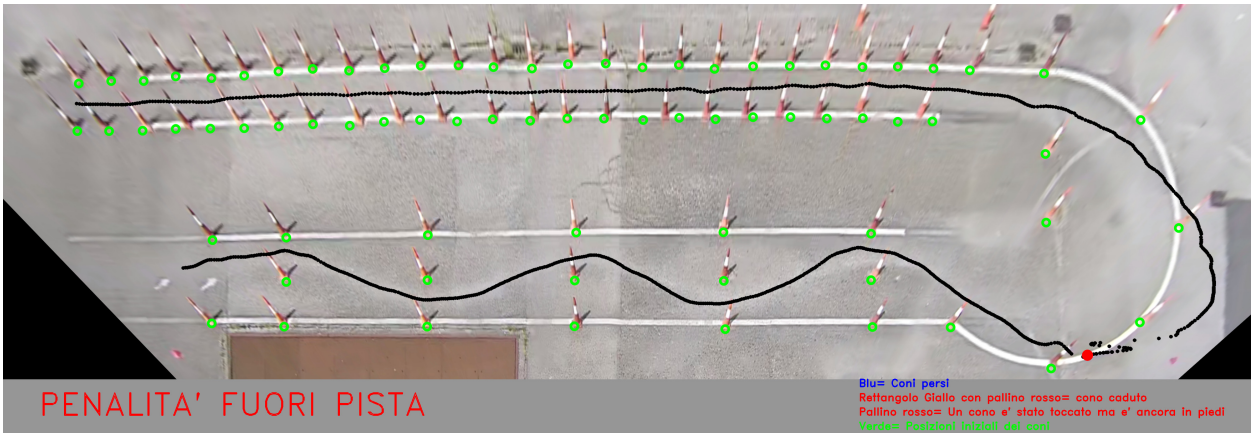



Figura 5.44: Traccia finale della moto nel caso di percorso ”fuori pista”. Il pallino rosso indica la posizione dove avviene l'uscita dalla pista; nella barra informativa in basso, è fornita la descrizione del tipo di penalità.

| | Esaminatore | Esaminato | Punteggio |
|---|---|---|---|
| Area percorso | 202.368 $m^2$ | 208.725$m^2$ | 0.969544 |
| Coefficiente correlazione x | | 0.974724 | 0.974724 |
| Coefficiente correlazione y | | 0.998567 | 0.998567 |
| Tempo | 18.5602 s | 19.9003 s | 0.932663 |
| Velocità massima | 571.41 cm/s | 718.886 cm/s | 0.79486 |
| Velocità minima | 27.10 cm/s | 13.06 cm/s | 0.658718 |
| Velocità media | 340.20 cm/s | 331.367 cm/s | 0.974668 |
| Deviazione Standard velocità | 100.36 cm/s | 98.3218 cm/s | 0.980135 |
| Punteggio finale | | | |

Tabella 5.8: Punteggi del percorso ”fuori pista”, confrontato con il percorso di riferimento.

Analizzando la tabella 5.8 si può notare come i punteggi del percorso sotto sia molto più alti rispetto ai punteggi ottenuti per i percorsi precedenti, questo perché la traiettoria in generale è molto simile a quella di un percorso ottimo (tranne per la parte di uscita dalla pista), ed il tempo e le velocità varie sono anche molto simili.

# Capitolo 6

# Conclusioni e Lavori Futuri

In questa relazione si è presentato il lavoro effettuato durante il tirocinio conclusivo della laurea triennale in Informatica della Università di Pisa. Il tirocinio si è svolto presso lo ISTC-CNR di Pisa nell'ambito del progetto di ricerca AI-RIDE: tale progetto mira ad automatizzare la valutazione dell'esame di guida per la patente dei motocicli per mezzo di un sistema informatico basato su Computer Vision e Intelligenza Artificiale. Alla partenza del tirocinio era già presente il sito di test completo di telecamere per l'acquisizione dei video e la prima parte di elaborazione dati inerente il riconoscimento degli elementi principali ovvero i coni segnaletici e il motociclo. L'obiettivo del tirocinio, partendo da questi dati, è stato la realizzazione delle funzionalità di ricostruzione dello scenario basandosi sulla posizione dei coni segnaletici e di tracciamento del percorso della moto durante l'esame di guida calcolando i parametri di tempo e velocità e segnalando le penalità previste. Inoltre si è teorizzato ed implementato un sistema di punteggio per giudicare una eventuale guida irregolare e si è realizzata una riproduzione virtuale della prova in ambiente grafico 3D. Per la progettazione di tali funzionalità è stato necessario apprendere la teoria su proiezioni prospettiche, filtro di Kalman e funzioni matematiche spline utilizzando algebra matriciale e concetti di fisica di base. Il software è stato sviluppato con il linguaggio C++ utilizzando le librerie opensource OpenCV e ALGLIB.
Complessivamente, dopo quanto scritto nei capitoli precedenti, e dall'analisi degli esperimenti mostrati nel paragrafo 5.8, si può evincere che la gran parte degli obiettivi inseguiti siano stati raggiunti con un buon grado di soddisfazione, soprattutto in relazione al tempo impiegato. Attualmente il sistema è in grado di individuare e specificare le caratteristiche di un test di guida ed in particolare:

- se l'esaminato ha eseguito il percorso correttamente, evidenziando i punteggi calcolati mediante il confronto con un percorso ottimo di riferimento;
- se l'esaminato ha toccato un cono, ha sbagliato lo slalom oppure è uscito di pista, segnalando la penalità e il punto in cui l'evento avviene;
- se l'esaminato ha percorso la pista troppo lentamente o velocemente, calcolando e visualizzando il tempo di percorrenza e segnalando l'eventuale penalità definita in ([5]);

- se l'esaminato ha (probabilmente) poggiato un piede a terra, riconoscendo che il veicolo si è fermato per un istante durante il percorso;

Nonostante tutti gli obiettivi prefissati siano stati raggiunti con buoni risultati, c'è ancora ampio margine di miglioramento. Tra i possibili sviluppi futuri si indica:

- una sperimentazione più ampia del sistema di punteggio di un percorso perché, come già scritto nella sezione 5.4, abbiamo avuto pochi esempi per validare correttamente l'idea proposta.

- utilizzare una spline bidimensionale piuttosto che una monodimensionale per la rappresentazione finale del percorso, evitando quindi la divisone dello stesso in tre zone distinte (si faccia riferimento alla sezione 5.3)

- un miglioramento del calcolo dei punteggi relativi a velocità e accelerazioni come diretta conseguenza del punto precedente: una diminuzione del rumore di partenza porterebbe a valori più rappresentativi della realtà.

Una parte del lavoro di tirocinio che non è stata sviluppata a sufficienza, ma che ha discrete potenzialità in termini di impatto, riguarda la realizzazione del Digital Twin in grafica 3D, ovvero della virtualizzazione della prova di guida per mezzo del motore grafico Unreal Engine. Al termine della stesura di questa tesi, questa parte del sistema è ancora molto sperimentale. Ad esempio la procedura soffre la presenza di dati troppo rumorosi; nonostante questo fornisce già dei buoni risultati come la possibilità di valutare le uscite di pista o il tocco di un cono in base alla dimensioni della moto pur partendo dall'informazione su un solo punto rappresentativo del veicolo. Questa caratteristica può ulteriormente essere migliorata utilizzando un modello in scala della moto al posto di quello generico usato attualmente.
Per concludere, si segnala che un articolo riguardante il progetto AI-RIDE è stato sottomesso alla 17-esima conferenza internazionale su SIGNAL IMAGE TECHNOLOGY e INTERNET BASED SYSTEMS [43].

# Appendice A

# Software sviluppato

Il codice sviluppato nel lavoro di tesi si occupa principalmente dei seguenti aspetti:

1. proiezioni da applicare al percorso della moto (estratto da un video), per ottenere delle posizioni con sistema di riferimento nel mondo reale;
2. tracking del percorso ottenuto al punto 1 utilizzando il Kalman Filter, per diminuire la presenza di rumori, ed inoltre come misura di sicurezza nel caso in cui il sistema di rilevamento degli oggetti perda qualche frame e quindi qualche passaggio della moto;
3. calcolo della spline utilizzata per ottenere una funziona che descrive il percorso fatto dalla moto e per approssimare in modo più regolare l'output del Kalman Filter;
4. analisi della traccia ottenuta al punto 3, sulla correttezza del percorso fatto, in relazione ad un altro percorso di riferimento fatto da un istruttore;

Nelle righe precedenti si fa riferimento al termine di traccia; questa traccia non è altro che un vettore di posizioni $(x, y, z)$ con la z sempre uguale a 0, visto che tutti gli eventi interessanti per noi avvengono ad altezza 0.

## A.1 Struttura della libreria

Il software è dotato della seguente struttura:

- nella directory principale troviamo il file "CMakeLists.txt" che contiene la lista di tutti i file .cpp che è necessario compilare.
- Sempre nella directory principale si trovano i seguenti script:
  - "compileProgram.sh". Esegue i comandi necessari per compilare il programma, e se necessario crea la directory output nella quale verrà inserito tutto l'output del programma.

  - ”runProgram.sh”. Script che esegue il programma, e da la possibilità di copiare tutto l’output dell’esecuzione, in una cartella nella directory immediatamente precedente a quella del software, dandogli come nome il primo argomento passato.
  - ”CleanFolder.sh”. Script che permette velocemente di ripulire la cartella output, prima di una possibile ri-esecuzione del software.

- Inoltre, sempre nella directory principale ci sono una serie di cartelle che sono:
  - ”3rdParty”. Questa cartella contiene tutti i .ccp .h, necessari a far funziona la libreria ”ALGLIB” ([39]), utilizzata per generare la spline del percorso.
  - ”include”. Contiene tutti gli header del software suddivisi per fase in cui operano.
  - ”ini”. Contiene tutti i file necessari per l’inizializzazione del software. Ad esempio il file ”airide.ini” in cui è possibile decidere quali percorsi processare, sia per l’esaminatore che per l’esaminato. Due .txt nella quale si trovano le matrici di Omografia pre-calcolate, ed una serie di .jpg utilizzate p come base della maggior parte dell’output generato.
  - ”src”. Contiene tutto il codice del programma.
  - ”trafficCones”. Contiene al suo interno diverse cartelle, una per ogni esempio di percorso. Ognuna delle cartelle è praticamente l’input del nostro programma, e al suo interno contiene due file ”eventiL1.txt” e ”eventiL2.txt” che descrivono rispettivamente tutte le info sui coni per la telecamera L1 e L2.
  - ”videoMoto”. Contiene al suo interno diverse cartelle, una per ogni esempio di percorso. Ognuna delle cartelle è praticamente l’input del nostro programma, e al suo interno contiene due file ”motoL1.txt” e ”motoL2.txt” che descrivono rispettivamente il percorso della moto per la telecamera L1 e L2.

## A.2 Istruzioni per l’esecuzione

Si è realizzato un repository su github dove si trova il codice sviluppato. La serie di comandi da eseguire per usare questo software è la seguente:

1. eseguire il comando ”git clone https://github.com/AIrideCNR/multiCameraProcessing.git”. Comando necessario per scaricare le repository da github, oppure scaricare il pacchetto zip.
2. recarsi nella cartella multiCameraProcessing sul vostro computer;
3. eseguire il comando ”bash compileProgram.sh”.
4. Impostare i dati di vostro interesse nel file ”./ini/airide.ini”. Sono già forniti diversi esempi nelle cartelle ”traffisCones” per i coni, e ”videoMoto” per i percorsi della moto.

5. Tornare nella cartella principale ed eseguire il comando ”bash runProgram.sh” per eseguire il software.

6. Troverete l’intero output del programma nella cartella output.

# Bibliografia


[1] Giuseppe Riccardo Leone, Marco Righi, Davide Moroni, and Francesco Paolucci. Towards Multi-Camera System for the Evaluation of Motorcycle Driving Test. In *PRELUDE 2022-International Workshop on PeRvasive sEnsing and muLtimedia UnDErstanding-In conjunction with SITIS 2022*, 2022.

[2] VEDLIot. Ai_ride ,aritficial intelligence, drive riding distributed eye. https://vedliot.eu/project/ai_ride/. Online; Controllata il 20/6/2023.

[3] CNIT Consorzio Nazionale Interuniversitario per le Telecomunicazioni. Airide demo event, june 14. https://www.cnit.it/en/2023/06/16/vedliot-h2020-ict-56-european-project-ai-ride-demo/. Online; Controllata il 20/6/2023.

[4] VEDLIot. Airide demo event, june 14. https://vedliot.eu/ai_ride-demo-event-june-14/. Online; Controllata il 18/6/2023.

[5] Ministero dei trasporti. "Decreto 26/09/2018 - prove di valutazione per conseguimento patenti a1, a2 e a". www.gazzettaufficiale.it/eli/id/2019/03/19/19A01769/sg. Online; Controllata il 7/5/2023.

[6] MotoPinas. "Singapore, using AI for motorcycle driving test". https://www.motopinas.com/motorcycle-news/singapore-using-ai-for-motorcycle-driving-tests.html. Online; Controllata il 7/5/2023.

[7] Glenn Jocher. YOLOv5 by Ultralytics. https://github.com/ultralytics/yolov5, 2020.

[8] OpenCV. Camera Calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Online; Controllata il 17-Aprile-2023.

[9] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.

[10] P. Baldi. *Introduzione alla probabilità. Con elementi di statistica*, chapter 1 "Statistica descrittiva". Collana di istruzione scientifica. McGraw-Hill Companies, 2003.

[11] D. Halliday, J. Walker, and R. Resnick. *Fondamenti di fisica.* Number v. 1 in Fondamenti di fisica. CEA, 2015.

[12] M. Luise and G.M. Vitetta. *Teoria dei segnali.* Collana di istruzione scientifica. McGraw-Hill Companies, 2009.

[13] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*, chapter 11.2.2 "Homgraphy", pages 555–558. Cengage Learning, 2014.

[14] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*, chapter 11.2.3 "Estimating homography from point correspondences", pages 558–561. Cengage Learning, 2014.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[16] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[17] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[18] Mohinder Grewal and Angus Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives. *Control Systems, IEEE*, 30:69 – 78, 07 2010.

[19] David Gaylor and E. Glenn Lightsey. *GPS/INS Kalman Filter Design for Spacecraft Operating in the Proximity of International Space Station.*

[20] B.D. Bojanov, H. Hakopian, and B. Sahakian. *Spline Functions and Multivariate Interpolations.* Mathematics and Its Applications. Springer Netherlands, 2014.

[21] Dario Andrea Bini. Interpolazione polinomiale a tratti. https://people.dm.unipi.it/bini/Didattica/IAN/appunti/splinenuovo.pdf. Accessed: 2023-03-28.

[22] Donald Fink, editor. *Color Television Standards: Selected Papers and Records of the NTSC.* McGraw-Hill, 1955.

[23] SMPTE. St 12-1:2008 - smpte standard - for television — time and control code. *St 12-1:2008*, page 1–40, February 2008. "When drop-frame compensation is applied to an NTSC television time code, the total deviation accumulated after one hour is reduced to approximately 3.6 ms. The total deviation accumulated over a 24-hour period is approximately 2.6 frames ( 86 ms).".

[24] Synchronisation and smpte timecode (time code). http://www.philrees.co.uk/articles/timecode.htm. Retrieved 2023-08-31.

[25] John Ratcliff. *Timecode: A User's Guide.* Focal Press, second edition, 1999.

[26] Charles Poynton. *A Technical Introduction to Digital Video.* John Wiley & Sons, 1996.

[27] Richard Szelinsky. Computer Vision: Algorithms and Applications, 2nd ed. https://szeliski.org/Book/, 2022.

[28] S.J.D. Prince. *Computer Vision: Models Learning and Inference.* Cambridge University Press, 2012.

[29] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision.* Cengage Learning, 2014.

[30] Azure AI. Ocr - optical character recognition. https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-ocr. Online; Controllata il 1-settembre-2023.

[31] Viso.ai. Ocr - optical character recognition. https://viso.ai/computer-vision/optical-character-recognition-ocr/. Online; Controllata il 1-settembre-2023.

[32] Wingfield Nick. Inside amazon go, a store of the future. https://www.nytimes.com/2018/01/21/technology/inside-amazon-go-a-store-of-the-future.html. Online; Controllata il 1-settembre-2023.

[33] Eric Guizzo. Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45:26 – 34, 08 2008.

[34] IEEE Spectrum. Covariant uses simple robot and gigantic neural net to automate warehouse picking. https://spectrum.ieee.org/covariant-ai-gigantic-neural-network-to-automate-warehouse-picking. Online; Controllata il 1-settembre-2023.

[35] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[36] Doug Roble. Vision in film and special effects. *SIGGRAPH Comput. Graph.*, 33(4):58–60, nov 1999.

[37] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[38] Epic Games. Unreal Engine 5. https://www.unrealengine.com/en-US/unreal-engine-5. Online; Controllata il 20-Agosto-2023.

[39] Sergey Bochkanov. ALGLIB. https://www.alglib.net/interpolation/spline3.php. Online; Controllata il 30-Agosto-2023.

[40] shutterstock. 3d models for professionals. https://www.turbosquid.com/. Accessed: 2023-08-12.

[41] Epic Games. Unreal Engine, Comparing Blueprints and C++ Use Cases. https://dev.epicgames.com/community/learning/tutorials/qM2K/unreal-engine-comparing-blueprints-and-c-use-cases. Online; Controllata il 20-Agosto-2023.

[42] Epic Games. Unreal Engine 5, Documentation. https://docs.unrealengine.com/5.0/en-US/. Online; Controllata il 20-Agosto-2023.

[43] Giuseppe Riccardo Leone; Marco Righi; Davide Moroni; Anthony Baiamonte; Davide Bulotta e Francesco Paolucci. Ai-ride: A multi-camera system for the evaluation of motorcycle driving test. *Conferenza SITIS*, 2023. Submitted to the 17TH INTERNATIONAL CONFERENCE ON SIGNAL IMAGE TECHNOLOGY and INTERNET BASED SYSTEMS.